

Natural

Referenzhandbuch

Bestellnummer: NAT316D030ALL

Dieses Handbuch gilt für Natural ab Version 3.1.6 für Großrechner, Version 5.1.1 für Windows und Version 5.1.1 für UNIX und OpenVMS.

Hierin enthaltene Beschreibungen unterliegen Änderungen und Ergänzungen, die in nachfolgenden Update-Serien oder Neuausgaben bekanntgegeben werden.

Anmerkungen und Verbesserungsvorschläge der Leserinnen und Leser sind sehr willkommen. Bitte richten Sie Ihre Anmerkungen an:

Software AG
Dokumentation
Uhlandstraße 12
64297 Darmstadt

Telefax: 06151-92-1612

© Juni 2002, Software AG

Alle Rechte vorbehalten

Printed in the Federal Republic of Germany

Software AG und/oder Software AG Produkte sind entweder Warenzeichen oder eingetragene Warenzeichen der Software AG. Andere hier erwähnte Produkte und Unternehmensnamen können Warenzeichen ihrer jeweiligen Eigentümer sein.

INHALTSVERZEICHNIS

VORWORT	1
Zu diesem Handbuch	1
Kapitel-Übersicht	2
Syntax-Symbole	2
Plattformspezifische Informationen	3
Deutsche Rechtschreibung	3
1. ALLGEMEINE INFORMATIONEN	5
Benutzervariablen	6
Namenskonventionen	7
Definition von Variablen	8
Statement-Referenzierung (r)	9
Definition von Format und Länge	11
Besondere Formate	13
Index-Notation	16
Referenzieren eines Datenbank-Arrays	20
Referenzieren des Internen Zählers für Datenbank-Arrays	24
Qualifizieren von Datenstrukturen	28
Konstanten	30
Numerische Konstanten	30
Alphanumerische Konstanten	32
Datums- und Zeitkonstanten	34
Hexadezimale Konstanten	37
Logische Konstanten	38
Gleitkomma-Konstanten	39
Attribut-Konstanten	40
Handle-Konstanten	40
Report-Spezifikation (rep)	41
Text-Notation	42
Kommentare	44
Ende eines Statements	44

Logische Bedingungen	45
Relationaler Ausdruck	47
Erweiterter Relationaler Ausdruck	51
MASK-Option	52
SCAN-Option	61
BREAK (Gruppenwechsel) in einer logischen Bedingung	63
IS-Option — Format-/Längenprüfung eines Wertes	65
Auswertung einer logischen Variablen	67
Modifizierte Kontrollvariablen	69
SPECIFIED-Option	71
Felder in logischen Bedingungen	72
Logische Operatoren in komplexen logischen Ausdrücken	74
Arithmetische Operationen	75
Initialisierung von Feldern	75
Datenübertragung	76
Abschneiden und Runden von Feldwerten	79
Format/Länge von Ergebnisfeldern bei arithmetischen Operationen	80
Arithmetische Operationen mit Gleitkomma-Zahlen	81
Arithmetische Operationen mit Datum und Zeit	83
Formatwahl im Hinblick auf die Verarbeitungszeit	86
Genauigkeit von Ergebnissen bei arithmetischen Operationen	87
Fehlerbedingungen bei arithmetischen Operationen	89
Verarbeitung von Arrays	89
Referenzen auf Sourcecode-Zeilenummern umnummerieren	96
2. SESSION-PARAMETER	97
Allgemeine Informationen	97
Setzen von Parameterwerten	98
Ändern von Parameterwerten auf Session-Ebene — Das GLOBALS-Kommando	98
Ändern von Parameterwerten auf Programm-Ebene — Das FORMAT-Statement	99
Ändern von Parameterwerten auf Statement-Ebene	99
Ändern von Parameterwerten auf Feld-Ebene	100
Verarbeitung von Parametern	101
Die Parameter im Überblick	102
AD — Attribut-Definition	104

AL — Alphanumerische Länge der Ausgabe	108
BX — Box-Definition (“Einrahmung”)	109
CC — Conditional Execution (Bedingte Programmausführung)	110
CD — Color Definition (Farbdefinition)	111
CF — Steuerzeichen für Terminalkommandos	112
CO — Compiler Output (Kompilier-Ausgabe)	113
CV — Control Variable (Kontrollvariable)	114
DC — Decimal Character (Dezimalstellenzeichen)	115
DF — Datumsformat	116
DFOUT — Datumsformat für Ausgabe	118
DFSTACK — Datumsformat für Stack	119
DFTITLE — Datumsformat in Standard-Seitenüberschrift	120
DU — Dump-Generierung	121
DY — Dynamische Attribute	123
EJ — Eject (Seitenvorschub)	125
EM — Editiermaske	126
ENDIAN — Endian-Modus für kompilierte Objekte	140
ES — Empty Line Suppression (Leerzeilenunterdrückung)	141
FC — Filler Character (Füllzeichen)	142
FCDP — Filler Character for Dynamically Protected Fields (Füllzeichen für dynamisch geschützte Felder)	143
FL — Floating Point Mantissa Length Fließpunkt-Mantissenlänge	144
FS — Format-Spezifikation	145
GC — Filler Character for Group Headers (Füllzeichen für Gruppenüberschriften)	146
HC — Header Centering (Überschriften-Zentrierung)	146
HE — Helproutine	147
HW — Heading Width (Überschriftenbreite)	150
IA — INPUT Assign Character (INPUT-Zuweisungszeichen)	151
IC — Insertion Character (Einfügungszeichen)	152
ID — INPUT-Delimiterzeichen	153

Natural Referenzhandbuch

IM — INPUT-Modus	154
IP — INPUT Prompt (Eingabeaufforderungstext)	155
IS — Identical Suppress (Unterdrücken identischer Werte)	156
KD — Key Definition (PF-Tasten-Anzeige)	157
LC — Leading Characters (Vorangestellte Zeichen)	158
LE — Limit Error Processing (Limit-Überschreitung)	159
LS — Line Size (Zeilenlänge)	160
LT — Limit of Records Read (Limit für Verarbeitungsschleifen)	161
MC — Multiple-Value Field Count (Anzahl multipler Feldwerte)	162
ML — Msg Line Position (Meldungszeilen-Position)	163
MP — Maximum Pages (Maximale Seitenzahl)	164
MS — Manual Skip (Manuelle Cursor-Positionierung)	165
MT — Maximum Time (Maximale CPU-Zeit)	166
NC — Use of Natural System Commands (Verwendung von Systemkommandos)	167
NL — Numerische Länge der Ausgabe	168
OPF — Overwriting of Protected Fields by Helproutines (Überschreiben geschützter Felder durch Helproutinen)	169
PC — Periodic Group Count (Anzahl der Periodengruppen-Ausprägungen) . . .	170
PD — NATPAGE Page Dataset	171
PM — Print Mode (Druck-Modus)	172
PS — Page Size (Seitenlänge)	174
REINP — Interner REINPUT bei ungültigen Daten	175
SA — Sound Terminal Alarm (Terminal-Warnton)	175
SF — Spacing Factor (Spaltenabstand)	176
SG — Sign Position (Vorzeichen-Stelle)	177
SL — Sourcecode-Zeilenlänge	177
SM — Structured Mode	178
SYMGEN — Generate Symbol Table (Symboltabelle generieren)	179
TC — Trailing Characters (Nachgestellte Zeichen)	180
TS — Konvertierung von Systemdatei-Programmausgaben	180

UC — Underlining Character (Unterstreichungszeichen)	181
WH — Warten auf Datensatz im “Hold”	182
ZD — Zero Division (Teilung durch Null)	183
ZP — Zero Printing (Anzeige von Nullwerten)	184
3. SYSTEMVARIABLEN	185
Alphabetische Liste der Variablen	186
*APPLIC-ID	187
*APPLIC-NAME	187
*COM	188
*CONTROL	189
*CONVID	189
*COUNTER	190
*CPU-TIME	190
*CURS-COL	191
*CURS-FIELD	191
*CURS-LINE	192
*CURSOR	193
*DATA	193
*DEVICE	194
*DIALOG-ID	194
*ERROR-LINE	195
*ERROR-NR	195
*ERROR-TA	196
*ETID	196
*EVENT	197
*GROUP	197
*HARDCOPY	198
*HARDWARE	198
*HOSTNAME	198
*INIT-ID	199

Natural Referenzhandbuch

*INIT-PROGRAM	200
*INIT-USER	200
*ISN	202
*LANGUAGE	203
*LENGTH	206
*LEVEL	206
*LIBRARY-ID	206
*LINE-COUNT	207
*LINESIZE	207
*LOG-LS	208
*LOG-PS	208
*MACHINE-CLASS	208
*NATVERS	209
*NET-USER	209
*NUMBER	210
*OCCURRENCE	211
*OPSYS	213
*OS	214
*OSVERS	214
*PAGESIZE	215
*PAGE-NUMBER	215
*PARAM-USER	216
*PATCH-LEVEL	216
*PF-KEY	217
*PF-NAME	218
*PID	218
*PROGRAM	218
*ROWCOUNT	219
*SCREEN-IO	219
*SERVER-TYPE	220
*STARTUP	221

*STEPLIB	222
*SUBROUTINE	223
*THIS-OBJECT	223
*TPSYS	224
*UI	224
*USER	225
*USER-NAME	225
*WINDOW-LS	225
*WINDOW-POS	226
*WINDOW-PS	226
*WINMGR	227
*WINMGRVERS	227
Datums- und Zeit-Systemvariablen	228
4. GROSSE UND DYNAMISCHE VARIABLEN/FELDER	231
Allgemeines	231
Definition dynamischer Variablen	233
Systemvariable *LENGTH(Feld)	234
Speichergrenzen-Prüfungen	235
Statements EXPAND und REDUCE	236
Verwendung dynamischer Variablen	237
5. SYSTEMFUNKTIONEN	259
Allgemeines	260
Systemfunktionen im SORT GIVE FUNCTIONS-Statement	261
Arithmetischer Überlauf bei AVER, NAVER, SUM oder TOTAL	261
Statement-Referenzierung (r)	261
ALPHABETISCHE LISTE DER SYSTEMFUNKTIONEN	262
AVER(r)(Feld)	262
COUNT(r)(Feld)	262
MAX(r)(Feld)	263
MIN(r)(Feld)	263

NAVER(r)(Feld)	263
NCOUNT(r)(Feld)	263
NMIN(r)(Feld)	264
OLD(r)(Feld)	264
SUM(r)(Feld)	264
TOTAL(r)(Feld)	265
Mathematische Funktionen	270
POS — Feldidentifikationsfunktion	274
POS und *CURS-FIELD	275
RET — Returncode-Funktion	276
SORTKEY — Sortierschlüssel-Funktion	277
6. TERMINALKOMMANDOS	279
Terminalkommandos – Übersicht	279
Was sind Terminalkommandos?	280
Eingabe eines Terminalkommandos	281
Verwendung von Terminalkommandos in Programmen	281
Verwendete Begriffe	282
%? — Hilfe-Informationen für Terminalkommandos	282
Terminalkommando-Übersicht	283
Nach Funktion eingeteilte Terminalkommandos	286
Terminalkommando-Tastenbelegungen	290
CLEAR-Taste — Aktive Operation unterbrechen	291
CTRL+D-Tasten — Aktive Operation unterbrechen	291
RESET+ENTER-Tasten — Aktive Schleife abbrechen	291
% — Fortsetzungsanzeiger für INPUT im Batch	292
%% und %. — Aktive Operation unterbrechen	293
%* — Anzeige von Eingabezeichen unterdrücken	296
Kommando-Syntax	296
%.P — Obersten Stack-Eintrag löschen	297
%.S — Stack-Daten lesen ohne zu löschen	297
%/ — End-of-File	298

%+ und %- — Natural Connection	299
%<TECH — Technische Informationen anzeigen	300
Kommando-Syntax	300
%<TEST — Debugging-Funktion aufrufen	300
%= — Zuordnen von Farben zu Feldern	301
Kommando-Syntax	301
%A — Ausführen eines Recordings	304
%B — Aktivieren/Deaktivieren des Recording-Prozesses	305
%B= — Library für Recording	306
%C — Seitenpuffer kopieren	307
Kommando-Syntax	307
%CS und %CC — Daten nach Stack/*COM kopieren	308
Kommando-Syntax	308
%D — Aktivieren des Keyword/Delimiter-Modus	310
%D= — “Outlining” steuern	310
%E — Mit NATPAGE aufgezeichnete Schirme anzeigen	312
%E= — Fehlerbehandlung ein-/ausschalten	314
Kommando-Syntax	314
%F — Aktivieren des Forms/Screen-Modus	314
%F — Aktivieren des Forms/Screen-Modus	315
%F= — Zeichen für Bildschirmfenster-Rahmen	316
%G — Ausführmodus für Recording	317
%H — “Hardcopy”-Ausgabe	318
%I — Aktuellen Schirm mit NATPAGE aufzeichnen	321
%J — Helproutine aufrufen	322
Kommando-Syntax	322
%KN/%KO/%KS — Siemens-Funktionstasten-Logik	323
%K und %KP — Simulieren von PF- und PA-Tasten	324
Kommando-Syntax	324
%L - Keine Umsetzung von Klein- in Großbuchstaben	325
Kommando-Syntax	325

%L= — Sprachcode	325
Kommando-Syntax	325
%M — Steuerung der Meldungszeile	326
%MSGSF — Anzeigeformat von Systemfehlermeldungen	328
%N — Aktivieren des “Non-Conversational”-Modus	329
Kommando-Syntax	329
%O — Beenden von NATPAGE	330
%P – NATPAGE für nachfolgende Schirme aktivieren	331
%P= — CALL-Optionen	332
%Q – Map-Ausdruck im Batch-Betrieb unterdrücken	336
Kommando-Syntax	336
%QO — Pseudo-konversationale Ausgabe unterdrücken	336
%QS — Gleichzeitige Ausgabe mehrerer Schirme	337
Kommando-Syntax	337
%R — INPUT-Statement wiederholen	338
Kommando-Syntax	338
%RM – Schreibschutz von lichtstift-sensitiven Feldern	339
%RN — Bildschirmdaten-Komprimierung unterdrücken	340
%RO — Bildschirm-Optimierung ein-/ausschalten	341
Kommando-Syntax	341
%S — Fortsetzen von NATPAGE	343
%T — Cursor am Schirmanfang plazieren	344
Kommando-Syntax	344
%Tll/cc — Cursor in Zeile ll, Spalte cc plazieren	344
Kommando-Syntax	344
%T+/%T– — Cursor in geschützte Felder plazieren	345
%T* — Cursor außerhalb des Fensters plazieren	346
Kommando-Syntax	346
%T= — Terminalspezifische Converter-Routine	347
%TRE — Externe Trace-Funktion aktivieren/deaktivieren	348
%TRI — Interne Trace-Funktion aktivieren/deaktivieren	349
%U — Umsetzen von Klein- in Großbuchstaben	350
Kommando-Syntax	350

%V — Steuerung des Print-Modus	351
%W — Window-Verarbeitung	352
Kommando-Syntax	352
%X — Steuerung der Statistikzeile/Infoline	361
Kommando-Syntax	361
%Y — Steuerung der PF-Tastenleiste	365
Kommando-Syntax	365
%Z — Arbeitsbereich des Editors löschen	368
Kommando-Syntax	368
7. SCHLÜSSELWÖRTER UND RESERVIERTE WÖRTER	369
INDEX	377

VORWORT

Zu diesem Handbuch

Dieses Handbuch gilt für alle Plattformen, auf denen Natural eingesetzt werden kann.

Es enthält detaillierte Informationen, die Sie beim Schreiben von Natural-Anwendungen benötigen.

Kapitel-Übersicht

Kapitel 1 (Seite 5) enthält *allgemeine Informationen* zu verschiedenen Punkten, die im Zusammenhang mit den Natural-Statements relevant sind: Benutzervariablen und Konstanten, Report-Spezifikation, Text-Notation, logische Verarbeitungsbedingungen sowie arithmetische Operationen.

Kapitel 2 (Seite 97) beschreibt die *Natural-Session-Parameter*. Session-Parameter dienen dazu, bestimmte Faktoren zu beeinflussen, wie etwa Aspekte der Report-Formatierung, Ausgabe von Feldern, usw.

Kapitel 3 (Seite 185) beschreibt die *Natural-Systemvariablen*. Systemvariablen dienen dazu, bei der Ausführung eines Programms bestimmte Systeminformationen (z.B. Datum und Uhrzeit) zu erhalten.

Kapitel 4 (Seite 231) beschreibt große und dynamische Variablen und Felder. Ab Natural Version 4.1 für UNIX, Windows und OpenVMS verfügt der Benutzer über eine erweiterte Funktionalität bei der Verwendung großer Variablen, wodurch die vorhandenen Speichergrößenbeschränkungen beseitigt werden, und eine dynamische Belegung dieser Variablen zur Ausführungszeit ermöglicht wird.

Kapitel 5 (Seite 259) beschreibt die *Natural-Systemfunktionen*. Systemfunktionen können Sie für mathematische und statistische Zwecke nutzen.

Kapitel 6 (Seite 279) beschreibt die *Natural-Terminalkommandos*. Diese Kommandos dienen überwiegend zur Ausführung bestimmter terminal-bezogener Funktionen.

Kapitel 7 (Seite 369) enthält eine Liste aller *Natural-Schlüsselwörter* und für *Natural reservierten Wörter*.

Syntax-Symbole

In den Diagrammen, die die Syntax von Natural-Statements sowie Teilen von Statements beschreiben, werden zahlreiche Symbole verwendet. Diese Symbole sind im *Natural Statements-Handbuch* erklärt.

Plattformspezifische Informationen

Wo erforderlich, werden plattformspezifische Informationen in der aktuellen Dokumentation durch die folgenden Begriffe identifiziert:

Großrechner	Bezieht sich auf die Betriebssysteme OS/390, VSE/ESA, VM/CMS und BS2000/OSD sowie auf alle von Natural unter diesen Betriebssystemen unterstützten TP-Monitore.
OpenVMS	Bezieht sich auf das Betriebssystem OpenVMS.
UNIX	Bezieht sich auf alle von Natural unterstützten UNIX-Systeme.
Windows	<p>In der aktuellen Natural-Dokumentation bezieht sich “Windows” als Begriff auf die folgenden Betriebssysteme:</p> <p>In einer Natural-Entwicklungsumgebung:</p> <ul style="list-style-type: none"> • Microsoft Windows NT • Microsoft Windows 2000 <p>In einer Natural-Laufzeitumgebung:</p> <ul style="list-style-type: none"> • Microsoft Windows 98 • Microsoft Windows NT • Microsoft Windows 2000
SQL-Datenbanken	Bezieht sich auf alle von Natural unterstützten SQL-Datenbanksysteme.

Deutsche Rechtschreibung

Dieses Handbuch ist in der alten deutschen Rechtschreibweise abgefaßt.

ALLGEMEINE INFORMATIONEN

Das hier vorliegende Handbuch gilt für alle Plattformen, auf denen Natural eingesetzt werden kann. Es enthält detaillierte Informationen, die Sie beim Schreiben von Natural-Anwendungen verwerten können.

Dieses Kapitel enthält Informationen zu folgenden Themen, die im Zusammenhang mit Natural-Statements relevant sind:

- Benutzervariablen (Seite 6)
- Konstanten (Seite 30)
- Report-Spezifikation (Seite 41)
- Text-Notation (Seite 42)
- Kommentare (Seite 44)
- Ende eines Statements (Seite 44)
- Logische Bedingungen (Seite 45)
- Arithmetische Operationen (Seite 75).

Wichtiger Hinweis zu Syntax-Symbolen

In den die Syntax von Natural-Statements und Teilen von Natural-Statements beschreibenden Diagrammen werden mehrere Symbole verwendet. Diese Symbole sind in der *Natural Statements-Dokumentation* erläutert.

Benutzervariablen

Benutzervariablen können dazu verwendet werden, in einem Programm oder Unterprogramm Zwischenergebnisse zu speichern.

- Namenskonventionen
- Definition von Variablen
- Statement-Referenzierung
- Definition von Format und Länge
- Besondere Formate
- Index-Notation
- Referenzieren eines Datenbank-Arrays
- Referenzieren des internen Zählers für Datenbank-Arrays
- Qualifizieren von Datenstrukturen.

Namenskonventionen

Der Name einer Benutzervariablen darf 1 bis 32 Zeichen lang sein.

Anmerkung:

Sie können Variablennamen verwenden, die über 32 Stellen lang sind (zum Beispiel um in komplexen Anwendungen die Lesbarkeit der Programme durch längere, "sprechendere" Variablennamen zu erhöhen); allerdings sind nur die ersten 32 Stellen signifikant und müssen eindeutig sein, die restlichen Stellen werden von Natural ignoriert.

Der Name einer Benutzervariablen darf kein reserviertes Natural-Wort sein.

Innerhalb eines Natural-Programms sollten Sie für eine Benutzervariable und ein Datenbankfeld nicht den gleichen Namen verwenden, da es sonst zu falschen Referenzierungen kommen kann (siehe **Qualifizieren von Datenstrukturen**, Seite 28).

Der Name einer Benutzervariablen darf folgende Zeichen enthalten:

Zeichen	Erklärung
A – Z	Buchstaben (Klein- und Großbuchstaben)
0 – 9	Ziffern
–	Bindestrich
@	“At“-Zeichen
_	Unterstrich
/	Schrägstrich
\$	Dollar-Zeichen
§	Paragraphenzeichen
&	Kaufmännisches Und-Zeichen
#	Rautenzeichen
+	Plus-Zeichen (nur als erstes Zeichen erlaubt)

Das erste Zeichen eines Variablennamens muß eines der folgenden Zeichen sein:

- ein Großbuchstabe
- #
- +
- &

Falls das erste Zeichen ein “#”, “+” oder “&” ist, muß der Name aus mindestens einem weiteren Zeichen bestehen.

“+” darf nur bei applikationsunabhängigen Variablen (AIVs) und Variablen in einer Global Data Area als erstes Zeichen eines Namens verwendet werden. AIV-Namen *müssen* mit “+” anfangen.

“&” als erstes Zeichen eines Namens wird verwendet bei der dynamischen Source-Generierung (siehe RUN-Statement im *Natural Statements-Handbuch*) sowie als dynamisch ersetzbares Platzhalterzeichen in Processing Rules (siehe Map-Editor-Beschreibung in Ihrem *Natural-Benutzerhandbuch* bzw. *User’s Guide*).

Definition von Variablen

Die Charakteristika einer Variablen definieren Sie mit folgender Notation:

(r,Format-Länge/Index)

Diese Notation steht hinter dem Variablennamen. Zwischen Name und Klammer dürfen ein oder mehrere Leerzeichen stehen. Zwischen den einzelnen Angaben innerhalb der Klammer dürfen keine Leerzeichen stehen. Sie können je nach Bedarf eine oder mehrere der Angaben machen; wenn Sie mehrere Angaben machen, müssen Sie sie durch die oben gezeigten Zeichen voneinander trennen.

Achtung:

Im Structured Mode oder wenn ein Programm eine DEFINE DATA LOCAL-Klausel enthält, können Variablen in einem Statement nicht dynamisch definiert werden. Dies gilt nicht für anwendungsunabhängige Variablen (AIVs).

Statement-Referenzierung (*r*)

Ein sogenanntes Statement-“Label” oder die Sourcecode-Zeilenummer kann dazu verwendet werden, auf ein vorhergehendes Statement zu verweisen (referenzieren). Dies können Sie einsetzen, wenn Sie statt Naturals Standard-Referenzierung (die bei jedem betroffenen Statement beschrieben ist) eine andere, eigene Referenzierung wünschen, oder zu Dokumentationszwecken.

Standard-Referenzierung von Datenbankfeldern

Wenn Sie keine Statement-Referenzierung angeben, gilt allgemein folgendes: Standardmäßig wird die innerste aktive Datenbankschleife (FIND, READ oder HISTOGRAM), mit der das betreffende Datenbankfeld gelesen wurde, referenziert. Wird das Feld in keiner aktiven Datenbankschleife gelesen, wird das letzte vorhergehende GET-Statement (bzw. im *Reporting Mode* auch FIND FIRST- oder FIND UNIQUE-Statement), mit dem das Feld gelesen wurde, referenziert.

Referenzierung über Statement-Labels

Ein Statement, das eine Verarbeitungsschleife auslöst und/oder einen Datenbankzugriff bewirkt, kann markiert werden, indem man ihm ein sogenanntes Label voranstellt. Ein mit einem Label markiertes Statement kann an späterer Stelle im Programm durch Angabe dieses Labels referenziert werden.

Das Label kann entweder in der Form *label.* vor oder in Klammern (*label.*) nach dem referenzierenden Objekt angegeben werden (aber nicht beides gleichzeitig).

Für Labels gelten die gleichen Namenskonventionen wie für Benutzervariablen. Der Punkt nach dem Label-Namen identifiziert das Label als Label.

Beispiel:

```
...  
RD. READ PERSON-VIEW BY NAME STARTING FROM 'JONES'  
  FD. FIND AUTO-VIEW WITH PERSONNEL-ID = PERSONNEL-ID (FD.)  
    DISPLAY NAME (RD.) FIRST-NAME (RD.) MAKE (FD.)  
  END-FIND  
END-READ  
...
```

Referenzierung über Sourcecode-Zeilennummern

Ein Statement kann auch referenziert werden, indem man die Nummer der Sourcecode-Zeile, in der das Statement steht, angibt.

Die Zeilennummer muß immer vierstellig (einschließlich vorangestellter Nullen) angegeben werden.

Beispiel:

```
...
0110 FIND EMPLOYEES-VIEW WITH NAME = 'SMITH'
0120   FIND VEHICLES-VIEW WITH MODEL = 'FORD'
0130     DISPLAY NAME (0110) MODEL (0120)
0140   END-FIND
0150 END-FIND
...
```

Weitere Informationen zum Referenzieren von Statements finden Sie im *Natural Leitfaden zur Programmierung*.

Definition von Format und Länge

Variablen fester Länge können mit den folgenden Formaten und entsprechenden Längen definiert werden:

Anmerkung:

Informationen zur Definition von Format und Länge von dynamischen Variablen finden Sie im Abschnitt “Definition of Dynamic Variables” in Ihrem Natural User’s Guide for Windows.

Format		Definierbare Länge (Anzahl der Stellen)	Interne Länge (in Bytes)
A	Alphanumerisch – auf Großrechnern: – auf allen anderen Plattformen	1 – 253 1 – 1073741824	1 – 253 1 – 1073741824
B	Binär – auf Großrechnern: – auf allen anderen Plattformen	1 – 126 1 – 1000000000	1 – 126 1 – 1000000000
C	Attribute Control = dynamische Attribute	–	2
D	Datum	–	4
F	Floating Point = Gleitkomma	4 oder 8	4 oder 8
I	Integer = Ganzzahl	1, 2 oder 4	1, 2 oder 4
L	Logisch	–	1
N	Numerisch (ungepackt)	1 – 29	1 – 29
P	Gepackt numerisch	1 – 29	1 – 15
T	Time = Zeit	–	7

Die Länge eines Feldes kann nur definiert werden, wenn gleichzeitig das Format definiert wird. Bei einigen Formaten erübrigt sich eine ausdrückliche Längendefinition (wie in obiger Tabelle gezeigt).

Bei Feldern, die das Format N oder P haben, können Sie die Länge auch in der Form “*nn.m*” definieren, wobei “*nn*” für die Stellen vor dem Komma (Dezimalpunkt) und “*m*” für die Stellen nach dem Komma steht. “*m*” darf nicht größer als 7 sein. “*nn*” und “*m*” dürfen zusammen nicht größer als 29 sein.

Anmerkung:

Im Reporting Mode erhält eine Benutzervariable, deren Format/Länge nicht explizit definiert wird, automatisch Format/Länge N7. Die Zuordnung dieser Standarddefinition kann mit dem Session-Parameter FS ein- bzw. ausgeschaltet werden.

Bei einem Datenbankfeld gelten die im DDM für das Feld definierten Format- und Längenangaben. (Im *Reporting Mode* ist es auch möglich, im Programm Format/Länge eines Datenbankfeldes anders zu definieren.)

Im *Structured Mode* muß die Format/Längen-Definition im DEFINE DATA-Statement oder einer Data Area erfolgen.

Beispiel für Format-/Längendefinition — Structured Mode:

```
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 FIRST-NAME
1 #NEW-SALARY (N6.2)
END-DEFINE
...
FIND EMPLOY-VIEW ...
...
COMPUTE #NEW-SALARY = ...
...
```

Im *Reporting Mode* darf die Format/Längen-Definition auch mitten im Programm erfolgen, wenn kein DEFINE DATA-Statement verwendet wird.

Beispiel für Format-/Längendefinition — Reporting Mode:

```
...
FIND EMPLOYEES ...
...
COMPUTE #NEW-SALARY(N6.2) = ...
...
```

Besondere Formate

Außer den normalen alphanumerischen (A) und numerischen (B, F, I, N, P) Formaten unterstützt Natural die Sonderformate C, D, T und L, die im folgenden beschrieben sind.

Format C (Attribute Control = dynamische Attributzuweisung)

Eine mit Format C definierte Variable kann dazu verwendet werden, einem in einem DISPLAY-, INPUT- oder WRITE-Statement benutzten Feld dynamisch Attribute zuzuweisen.

Bei einer Variablen mit Format C kann die Länge nicht definiert werden. Eine Format-C-Variablen erhält immer automatisch die Länge 2.

Beispiel:

```
DEFINE DATA LOCAL
1 #ATTR(C)
1 #A(N5)
END-DEFINE
...
MOVE (AD=I CD=RE) TO #ATTR
INPUT #A (CV=#ATTR)
...
```

Weitere Informationen siehe Session-Parameter CV.

Formate D (Date = Datum) und T (Time = Zeit)

Mit den Formaten D bzw. T definierte Variablen können für Datum- und Uhrzeitanzeigen und -berechnungen verwendet werden. Format D kann nur Datumsinformationen enthalten, während Format T Datums- *und* Zeitinformationen enthalten kann, denn Datumsinformationen sind ein Bestandteil von Zeitinformationen. Die Zählinheit für Zeit ist Zehntelsekunden.

Datums- und Zeitvariablen werden ohne Längenangaben definiert. Eine Datumsvariable erhält automatisch eine Länge von 4 Bytes (P6), eine Zeitvariable eine Länge von 7 Bytes (P12).

Beispiel:

```
DEFINE DATA LOCAL
1 #DAT1 (D)
END-DEFINE
*
MOVE *DATX TO #DAT1
ADD 7 TO #DAT1
WRITE '=' #DAT1
END
```

Weitere Informationen siehe Session-Parameter EM sowie Systemvariablen *DATX und *TIMX.

Der Wert in einem Datumsfeld muß im Bereich vom 1. Januar 1582 bis 31. Dezember 2699 liegen.

Format L (Logisch)

Eine mit Format L definierte Variable kann für eine logische Bedingung verwendet werden; sie kann entweder den Wert “T” (True = wahr) oder “F” (False = falsch) enthalten.

Eine logische Variable wird ohne Längenangabe definiert; sie erhält automatisch die Länge 1.

Beispiel:

```
DEFINE DATA LOCAL
1 #SWITCH(L)
END-DEFINE
MOVE TRUE TO #SWITCH
...
IF #SWITCH
...
MOVE FALSE TO #SWITCH
ELSE
...
MOVE TRUE TO #SWITCH
END-IF
```

Weitere Informationen über die Darstellung logischer Werte siehe Session-Parameter EM.

Format “Handle”

Eine als “HANDLE OF *dialog-element-type*” definierte Variable kann als “GUI-Handle” verwendet werden.

Eine als “HANDLE OF OBJECT” definierte Variable kann als “Objekt-Handle” (object handle) verwendet werden.

Nähere Informationen zu “GUI-Handles” siehe *Natural User’s Guide for Windows*.

Nähere Informationen zu “Objekt-Handles” siehe *NaturalX Documentation*.

Index-Notation

Für Felder, die ein Array darstellen, wird eine Index-Notation verwendet.

Für Index-Notationen kann eine ganzzahlige numerische Konstante oder Benutzervariable verwendet werden. Eine Systemvariable, Systemfunktion oder qualifizierte Variable kann in Index-Notationen nicht verwendet werden.

Definieren von Arrays — Beispiele:

1. **#ARRAY (3)**

Definiert ein eindimensionales Array mit drei Ausprägungen.

2. **FIELD (*label*,A20/5)** oder ***label*.FIELD(A20/5)**

Definiert ein Array mit fünf Ausprägungen von einem Datenbankfeld (alphanumerisches Format, Länge 20) unter Referenzierung des mit "*label*." markierten Statements.

3. **#ARRAY (N7.2/1:5,10:12,1:4)**

Definiert ein Array mit Format/Länge N7.2 und drei Dimensionen, und zwar mit 5 Ausprägungen in der ersten, 3 Ausprägungen in der zweiten und 4 Ausprägungen in der dritten Dimension.

4. **FIELD (*label*:*i* + 5)** oder ***label*.FIELD(*i* + 5)**

Definiert ein Array von einem Datenbankfeld, das über das mit "*label*." markierte Statement referenziert wird. FIELD stellt ein multiples Feld oder ein Feld aus einer Periodengruppe dar, wobei "*i*" den Index innerhalb der Datenbanksausprägung angibt. Die Größe des Arrays innerhalb des Programms ist mit 6 Ausprägungen (*i*:*i* + 5) definiert. Der Datenbank-Indexanfang ist als Variable angegeben, damit das Programm-Array innerhalb der Ausprägungen des multiplen Feldes bzw. der Periodengruppe positionieren kann. Für jede Repositionierung von "*i*" muß mittels eines GET- oder GET SAME-Statements ein neuer Datenbankzugriff durchgeführt werden.

Natural erlaubt die Definition von Arrays, deren Index nicht mit "1" beginnen muß. Zur Laufzeit prüft Natural, ob die bei der Referenzierung angegebenen Indexwerte nicht die Gesamtgröße der in der Definition angegebenen Dimensionen überschreitet.

Anmerkung:

Aus Gründen der Kompatibilität mit Natural Version 1 ist es bei der Angabe eines Array-Bereiches im Reporting Mode (außer in einem DEFINE DATA-Statement) auch erlaubt, statt des Doppelpunktes (:) einen Bindestrich (-) zu verwenden; beide Zeichen dürfen allerdings nicht vermischt werden.

Auf Großrechnern können Indexwerte im Bereich von -32767 bis +32767 liegen. Die größtmögliche Anzahl von Ausprägungen pro Array ist 32767. Die Maximalgröße eines Arrays insgesamt ist 32767 Bytes (= 32 KB - 1). Die Maximalgröße einer Data Area pro Programmierobjekt ist 16 777 215 Bytes (16 MB -1).

Auf allen anderen Plattformen ist der größte Indexwert 1 073 741 824. Pro Programmierobjekt liegt die Maximalgröße einer Data Area bei 1 073 741 824 Bytes (1 GB). Verwenden Sie den Profilparameter DSLM, um diese Beschränkungen aus Kompatibilitätsgründen auf die für Großrechner gültigen Beschränkungen zu reduzieren.

Bei der Index-Referenzierung können Sie einfache arithmetische Ausdrücke mit den Operatoren “+” und “-” verwenden. Wenn arithmetische Ausdrücke als Indexe verwendet werden, muß vor und nach den Operatoren “+” und “-” jeweils ein Leerzeichen stehen.

Arrays in Gruppenstrukturen werden von Natural Feld für Feld aufgelöst, und nicht Gruppenausprägung für Gruppenausprägung.

Beispiel für Auflösung von Arrays in Gruppen:

```
DEFINE DATA LOCAL
1 #GROUP (1:2)
  2 #FIELD A (A5/1:2)
  2 #FIELD B (A5)
END-DEFINE
...
```

Wenn die oben definierte Gruppe mit einem WRITE-Statement ausgegeben würde:

```
WRITE #GROUP (*)
```

würden die Ausprägungen in der folgenden Reihenfolge ausgegeben:

```
#FIELD A(1,1) #FIELD A(1,2) #FIELD A(2,1) #FIELD A(2,2) #FIELD B(1) #FIELD B(2)
```

und nicht:

```
#FIELD A(1,1) #FIELD A(1,2) #FIELD B(1) #FIELD A(2,1) #FIELD A(2,2) #FIELD B(2)
```

Referenzieren von Arrays — Beispiele:1. **#ARRAY (1)**

Referenziert die 1. Ausprägung eines eindimensionalen Arrays.

2. **#ARRAY (7:12)**

Referenziert die 7. bis 12. Ausprägung eines eindimensionalen Arrays.

3. **#ARRAY (i + 5)**

Referenziert die “i plus 5”-te Ausprägung eines eindimensionalen Arrays.

4. **#ARRAY (5,3:7,1:4)**

Hier wird ein dreidimensionales Array referenziert: und zwar die 5. Ausprägung in der ersten Dimension, die 3. bis 7. Ausprägung (also 5 Ausprägungen) in der zweiten Dimension und die 1. bis 4. Ausprägung (also 4 Ausprägungen) in der dritten Dimension.

5. Um *alle* Ausprägungen in einer Dimension zu referenzieren, kann ein Stern “*” verwendet werden:

```
DEFINE DATA LOCAL
1 #ARRAY1 (N5/1:4,1:4)
1 #ARRAY2 (N5/1:4,1:4)
END-DEFINE
...
ADD #ARRAY1 (2,*) TO #ARRAY2 (4,*)
...
```

Angabe eines Schrägstrichs vor einer Array-Ausprägung

Steht hinter einem Variablennamen eine vierstellige Zahl in Klammern, so interpretiert Natural diese als Zeilennummer-Referenzierung auf ein Statement. Daher muß vor einer vierstelligen Array-Ausprägung ein Schrägstrich “/” stehen, um sie als Array-Ausprägung kenntlich zu machen. Zum Beispiel:

#ARRAY(/1000)

nicht: **#ARRAY(1000)**

da letzteres als Referenzierung auf Sourcecode-Zeile 1000 interpretiert würde.

Falls ein Indexvariablen-Name fälschlicherweise als Format-/Längendefinition interpretiert werden könnte, muß der Index-Notation ein Schrägstrich “/” vorangestellt werden, um zu kennzeichnen, daß es sich um einen Index handelt. Ergibt sich die Ausprägung eines Arrays beispielsweise aus dem Wert der Variablen “N7”, muß die Ausprägung wie folgt angegeben werden:

#ARRAY (/N7)

nicht: **#ARRAY (N7)**

da letzteres fälschlicherweise als Definition eines numerischen, 7 Byte langen Feldes interpretiert würde.

Referenzieren eines Datenbank-Arrays

Referenzieren von multiplen Feldern und Periodengruppen

Bestimmte Werte eines multiplen Feldes oder eines in einer Periodengruppe enthaltenen Feldes, das in einem View/DDM definiert ist, können mit verschiedenen Index-Notationen definiert bzw. referenziert werden.

Die ersten 10 Werte sowie die "I" bis "I plus 10"-ten Werte eines multiplen Feldes/ Periodengruppenfeldes eines Datenbank-Satzes können beispielsweise wie folgt referenziert werden:

```
DEFINE DATA LOCAL
1 I(I2)
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 LANG (1:10)
  2 LANG (I:I + 10)
END-DEFINE
```

oder:

```
RESET I(I2)
...
READ EMPLOYEES
OBTAIN LANG(1:10) LANG(I:I + 10)
```

Anmerkung:

Pro Array darf nicht zweimal dieselbe untere Grenze des Indexbereichs angegeben werden (dies gilt für konstante wie für variable Indexe). Bei einem Array, das mit einem variablen Index definiert wird, muß die Untergrenze als die Variable selbst angegeben werden und die Obergrenze als die Summe aus der Variablen und einer Konstanten.

Referenzieren von mit Konstanten definierten Arrays

Ein mit Konstanten definiertes Array kann entweder über Konstanten oder über Variablen referenziert werden. Die Obergrenze des Arrays kann nicht überschritten werden. Wird eine Konstante verwendet, so prüft Natural bei der Kompilierung die Obergrenze.

```
RESET I(I2)
I = 1
READ EMPLOYEES
OBTAIN LANG(1:10)
WRITE LANG(1) / LANG(5:9) / LANG(1:10)
```

```
DEFINE DATA LOCAL
1 I(I2)
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 LANG (1:10)
END-DEFINE
*
READ EMPLOY-VIEW
  FOR I 1 TO 5
    WRITE LANG(1.I)
  END-FOR
END-READ
END
```

Wird ein multiples Feld bzw. Periodengruppenfeld mehrmals mit Konstanten definiert und soll mit Variablen referenziert werden, geschieht dies mit folgender Syntax:

```
DEFINE DATA LOCAL
1 I(I2)
1 J(I2)
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 LANG (1:5)
  2 LANG (11:20)
END-DEFINE
*
READ EMPLOY-VIEW
  FOR I 1 TO 2
    FOR J I TO 4
      DISPLAY 'LANGUAGE' LANG(1.I:J)
    END-FOR
  END-FOR
END-READ
END
```

Referenzieren von mit Variablen definierten Arrays

Multiple Felder/Periodengruppenfelder, die in Arrays mit Variablen definiert sind, müssen mit denselben Variablen referenziert werden.

```
RESET I(I2)
I = 1
READ EMPLOYEES
OBTAIN LANG(I:I+10)
WRITE LANG(I) / LANG (I+5:I+6)
END
```

Soll ein anderer Index verwendet werden, muß die erste angetroffene Definition des Arrays mit variablem Index eindeutig referenziert werden. Dies geschieht durch Qualifizierung des Indexausdrucks:

```
RESET I(I2) J(I2)
I = 1
J = 1
READ EMPLOYEES
OBTAIN LANG(I:I+10)
WRITE LANG(I.J) / LANG(I.1:5)
END
```

Mit "I." wird die Array-Definition eindeutig referenziert und gleichzeitig auf den ersten Wert innerhalb des gelesenen Array-Bereichs (LANG (I:I+10)) "positioniert".

Der aktuelle Inhalt von "I" zum Zeitpunkt des Datenbankzugriffs bestimmt die Anfangsausprägung des Datenbank-Arrays.

Referenzieren mehrmals definierter Arrays

Zwecks eindeutiger Referenzierung des gewünschten Array-Bereiches ist es bei mehrmals definierten Arrays in der Regel erforderlich, bei der Referenzierung den Indexausdruck mit anzugeben.

Beispiel:

```
DEFINE DATA LOCAL
  1 I(I2) INIT <1>
  1 J(I2) INIT <2>
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 LANG (1:10)
    2 LANG (5:20)
END-DEFINE
READ (2) EMPLOY-VIEW
  WRITE LANG(1.1:10) / LANG(5.5:15)
  DISPLAY LANG(1.I:I+2) / LANG(5.J)
END-READ
END
```

Bei der Definition von multiplen Feldern und Periodengruppenfeldern unter Verwendung von Indexvariablen wird eine ähnliche Syntax verwendet:

Beispiel:

```
DEFINE DATA LOCAL
  1 I(I2) INIT <1>
  1 J(I2) INIT <1>
  1 N(I2) INIT <1>
  1 M(I2) INIT <1>
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 LANG (I:I+10)
    2 LANG (J:J+5)
    2 LANG (1:2)
END-DEFINE
READ (2) EMPLOY-VIEW
  WRITE LANG(I.I) / LANG(I.I:I+10)
  DISPLAY LANG(J.N) / LANG(J.N:M)
  DISPLAY LANG(1.N) / LANG(1.N:M)
END-READ
END
```

Referenzieren des Internen Zählers für Datenbank-Arrays

Manchmal ist es erforderlich, ein multiples Feld und/oder eine Periodengruppe zu referenzieren, ohne die Anzahl der vorhandenen Werte bzw. Ausprägungen zu kennen. Adabas besitzt einen internen Zähler, der die Anzahl der Werte eines multiplen Feldes und die Anzahl der Ausprägungen einer Periodengruppe zählt. Diesen Zähler können Sie referenzieren, indem Sie unmittelbar vor dem betreffenden Feldnamen "C*" eingeben. Vgl. Data-Area-Editor-Zeilenskommando ".*" (wie in Ihrem *Natural-Benutzerhandbuch* bzw. *User's Guide* beschrieben).

Der Zähler gibt die Werte im Format N3 aus.

Beispiele:

C*LANG	Gibt die Anzahl der Werte des multiplen Feldes LANG aus.
C*INCOME	Gibt die Anzahl der Ausprägungen der Periodengruppe INCOME aus.
C*BONUS(1)	Gibt die Anzahl der Werte des multiplen Feldes BONUS in der ersten Ausprägung einer Periodengruppe aus (ausgehend von der Annahme, daß das multiple Feld BONUS Bestandteil einer Periodengruppe ist).

Hinweis für SQL-Datenbanken:

Für SQL-Datenbanken kann die C-Notation nicht verwendet werden.*

Hinweis für VSAM-Datenbanken:

Mit der C-Notation erhalten Sie nicht die Anzahl der Ausprägungen/Werte, sondern die Nummer der höchsten Ausprägung bzw. des höchsten Wertes (entsprechend der Definition im DDM (MAXOCC)).*

Beispiel für C*-Variable:

```

/* EXAMPLE 'ICOUNT':
/* USING C*NOTATION TO OBTAIN INTERNAL COUNT FOR DATABASE ARRAY
/*****
LIMIT 2
READ EMPLOYEES BY CITY
OBTAIN SALARY(1:5)
WRITE NOTITLE 'NAME:' NAME / 'NUMBER OF LANGUAGES SPOKEN:' C*LANG
          5X 'LANGUAGE 1:' LANG (1)
          5X 'LANGUAGE 2:' LANG (2)
/*****
WRITE 'SALARY DATA:'
FOR #A (N1) FROM 1 TO C*INCOME
  WRITE 'SALARY' #A SALARY (1.#A)
LOOP
/*****
WRITE 'THIS YEAR BONUS:' C*BONUS(1) BONUS (1,1) BONUS (1,2)
  / 'LAST YEAR BONUS:' C*BONUS(2) BONUS (2,1) BONUS (2,2)
SKIP 1
END

```

```

NAME: SENKO
NUMBER OF LANGUAGES SPOKEN:      1      LANGUAGE 1: ENG      LANGUAGE 2:
SALARY DATA:
SALARY 1      31500
SALARY 2      29900
SALARY 3      28100
SALARY 4      26600
SALARY 5      25200
THIS YEAR BONUS:      0      0      0
LAST YEAR BONUS:      0      0      0

NAME: GODEFROY
NUMBER OF LANGUAGES SPOKEN:      1      LANGUAGE 1: FRE      LANGUAGE 2:
SALARY DATA:
SALARY 1      170300
THIS YEAR BONUS:      1      50000      0
LAST YEAR BONUS:      0      0      0

```

C* für multiple Felder in Periodengruppen

Für ein multiples Feld *innerhalb* einer Periodengruppe können Sie ebenfalls eine C*-Variable mit Angabe eines Indexbereichs definieren.

Die folgenden Beispiele verwenden das multiple Feld BONUS, das Teil der Periodengruppe INCOME ist. Alle drei Beispiele führen zu dem gleichen Ergebnis.

Beispiel 1 — Reporting Mode:

```
READ EMPLOYEES BY PERSONNEL-ID FROM 11100117
  OBTAIN C*BONUS (1:3)
        BONUS (1:3,1:3)
*
  DISPLAY C*BONUS (1:3)
        BONUS (1:3,1:3)
END
```

Beispiel 2 — Structured Mode:

```
DEFINE DATA LOCAL
1 EMP VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 INCOME (1:3)
  3 C*BONUS
  3 BONUS (1:3)
END-DEFINE
READ EMP BY PERSONNEL-ID FROM 11100117
  DISPLAY C*BONUS (1:3)
        BONUS (1:3,1:3)
END-READ
END
```

Beispiel 3 — Structured Mode:

```
DEFINE DATA LOCAL
1 EMP VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 C*BONUS (1:3)
  2 INCOME (1:3)
  3 BONUS (1:3)
END-DEFINE
READ EMP BY PERSONNEL-ID FROM 11100117
  DISPLAY PERSONNEL-ID C*BONUS (*) BONUS (*,*)
END-READ
END
```

Anmerkung:

Da Adabas im Format Buffer keine Bereiche für Zähler-Felder erlaubt, werden diese als Einzelelemente generiert; daher kann die Angabe eines C-Indexbereiches für ein großes Array zu einem Überlauf des Adabas Format Buffers führen.*

Qualifizieren von Datenstrukturen

Um beim Referenzieren eines Feldes das Feld zu identifizieren, können Sie es qualifizieren; d.h. Sie geben vor dem Feldnamen den Namen des Level-1-Datenelements, in dem das Feld enthalten ist, gefolgt von einem Punkt an.

Wenn ein Feld durch seinen Namen allein nicht eindeutig identifiziert werden kann (zum Beispiel, wenn derselbe Feldname in mehreren Gruppen/Views verwendet wird), müssen Sie das Feld beim Referenzieren qualifizieren.

Die Kombination von Level-1-Datenelement und Feldname muß eindeutig sein.

Beispiel:

```
DEFINE DATA LOCAL
1 FULL-NAME
  2 LAST-NAME (A20)
  2 FIRST-NAME (A15)
1 OUTPUT-NAME
  2 LAST-NAME (A20)
  2 FIRST-NAME (A15)
END-DEFINE
...
MOVE FULL-NAME.LAST-NAME TO OUTPUT-NAME.LAST-NAME
...
```

Der Qualifizierer muß ein Level-1-Datenelement sein.

Beispiel:

```
DEFINE DATA LOCAL
1 GROUP1
  2 SUB-GROUP
    3 FIELD1 (A15)
    3 FIELD2 (A15)
END-DEFINE
...
MOVE 'ABC' TO GROUP1.FIELD1
...
```

Anmerkung:

Falls Sie für eine Benutzervariable und ein Datenbankfeld denselben Namen verwenden (was Sie ohnehin nicht tun sollten), müssen Sie das Datenbankfeld qualifizieren, wenn Sie es referenzieren wollen; sonst wird stattdessen die Benutzervariable referenziert.

Konstanten

Konstanten können überall in Natural-Programmen verwendet werden. In diesem Abschnitt werden die folgenden Arten von Konstanten und ihre Benutzungsweise behandelt:

- Numerische Konstanten
- Alphanumerische Konstanten
- Datums- und Zeitkonstanten
- Hexadezimale Konstanten
- Logische Konstanten
- Gleitkomma-Konstanten
- Handle-Konstanten

Numerische Konstanten

Eine numerische Konstante kann 1 bis 29 Stellen lang sein. Eine numerische Konstante, die in einem COMPUTE- oder MOVE-Statement oder einem arithmetischen Statement verwendet wird, darf ein Komma (Dezimalpunkt) und ein Vorzeichen enthalten.

Beispiele:

```
MOVE 3 TO #XYZ  
COMPUTE #PRICE = 23.34  
COMPUTE #XYZ = -103  
COMPUTE #A = #B * 6074
```

Anmerkung:

Intern werden numerische Konstanten ohne Nachkommastellen als Ganzzahl (Format I) dargestellt, während numerische Konstanten mit Nachkommastellen sowie numerische Konstanten ohne Nachkommastellen, die zu groß sind, um in Format I zu passen, in gepackter Form (Format P) dargestellt werden.

Auf Großrechnern werden numerische Konstanten intern in gepackter Form (Format P) dargestellt; Ausnahme: eine numerische Konstante in einer arithmetischen Operation, in der der andere Operand eine Ganzzahl-Variable (Format I) ist, wird als Ganzzahl (Format I) dargestellt.

Prüfung numerischer Konstanten

Wenn numerische Konstanten in einem der Statements MOVE, COMPUTE oder DEFINE DATA mit INIT-Option verwendet werden, prüft Natural bereits während der *Kompilierung*, ob eine Konstante in das betreffende Feld paßt. Dadurch werden Laufzeitfehler vermieden in Situationen, in denen eine derartige Fehlerbedingung bereits bei der Kompilierung erkannt werden kann.

Alphanumerische Konstanten

Eine alphanumerische Konstante kann 1 bis 253 alphanumerische Zeichen lang sein.

Eine alphanumerische Konstante muß entweder in Apostrophen (') oder in Anführungszeichen (") stehen.

Beispiele:

```
MOVE 'ABC' TO #FIELDX
```

```
MOVE '% INCREASE' TO #TITLE
```

```
DISPLAY "LAST-NAME" NAME
```

Eine alphanumerische Konstante, die dazu benutzt wird, einer Benutzervariablen einen Wert zuzuweisen, darf nicht über eine einzige Sourcecode-Zeile hinausgehen.

Apostrophe innerhalb alphanumerischer Konstanten

Ein Apostroph, das Teil einer in Apostrophen stehenden alphanumerischen Konstanten ist, muß entweder durch doppelte Apostrophe oder durch ein einzelnes Anführungszeichen dargestellt werden.

Ein Apostroph, das Teil einer in Anführungszeichen stehenden alphanumerischen Konstanten ist, wird als einzelnes Apostroph dargestellt.

Beispiel:

Um folgende Ausgabe zu erhalten:

```
HE SAID, 'HELLO'
```

können Sie jede der folgenden Notationen verwenden:

```
WRITE 'HE SAID, ''HELLO'''
```

```
WRITE 'HE SAID, "HELLO"'
```

```
WRITE "HE SAID, ""HELLO"""
```

```
WRITE "HE SAID, 'HELLO'"
```

Anmerkung:

Falls Anführungszeichen nicht wie oben gezeigt in Apostrophe umgesetzt werden, liegt dies an der Einstellung des Profilparameters TQ; Näheres hierzu erfahren Sie von Ihrem Natural-Administrator.

Verknüpfung alphanumerischer Konstanten

Mittels eines Bindestriches können mehrere alphanumerische Konstanten zu einem einzigen Wert verkettet werden.

Beispiele:

```
MOVE 'XXXXXX' -  
      'YYYYYY' TO #FIELD
```

```
MOVE "ABC" - 'DEF' TO #FIELD
```

Auf diese Weise können alphanumerische Konstanten auch mit hexadezimalen Konstanten verkettet werden.

Datums- und Zeitkonstanten

Eine Datumskonstante kann in Verbindung mit einer Variablen, die das Format D hat, verwendet werden. Datumskonstanten können folgende Datumsformate haben:

D'YYYY-MM-DD'	Internationales Datumsformat
D'DD.MM.YYYY'	Deutsches Datumsformat
D'DD/MM/YYYY'	Europäisches Datumsformat
D'MM/DD/YYYY'	USA-Datumsformat

Beispiel:

```
DEFINE DATA LOCAL
1 #DATE (D)
END-DEFINE
...
MOVE D'1997-04-27' TO #DATE
...
```

Das Standardformat für Datumsangaben wird vom Natural-Administrator mit dem Profilparameter DTFORM festgelegt.

Eine Zeitkonstante kann in Verbindung mit einer Variablen, die das Format T hat, verwendet werden. Eine Zeitkonstante hat folgende Form:

T'*hh:ii:ss***'**

Zeichen	Bedeutung
hh	Stunden
ii	Minuten
ss	Sekunden

Beispiel:

```
DEFINE DATA LOCAL
1 #TIME (T)
END-DEFINE
...
MOVE T'11:33:00' TO #TIME
...
```

Erweiterte Zeitkonstanten

Eine Zeitvariable (Format T) kann Datums- und Zeitinformationen enthalten, da Datumsinformationen einen Bestandteil von Zeitinformationen darstellen; mit einer “normalen” Zeitkonstanten (Präfix “T”) können jedoch nur die Zeitinformationen einer Zeitvariablen beeinflusst werden:

T’hh:ii:ss’

Mit einer erweiterten Zeitkonstanten (Präfix “E”) ist es möglich, den gesamten Inhalt einer Zeitvariablen — einschließlich der Datumsinformationen — zu beeinflussen:

E’yyyy–mm–dd hh:ii:ss’

Ansonsten entspricht die Verwendung einer erweiterten Zeitkonstanten in Verbindung mit einer Zeitvariablen der einer normalen Zeitkonstanten.

Anmerkung:

Das Format, in dem die Datumsinformationen in einer erweiterten Zeitkonstanten angegeben werden müssen, hängt von der Einstellung des Profilparameters DTFORM ab (vgl. Seite 34). Die obige Darstellung geht von DTFORM=I (internationales Datumsformat) aus.

Hexadezimale Konstanten

Eine hexadezimale Konstante dient dazu, einen Wert einzugeben, der nicht über den auf der Tastatur verfügbaren Zeichensatz eingegeben werden kann.

Eine hexadezimale Konstante wird durch den vorangestellten Buchstaben "H" gekennzeichnet. Die Konstante selbst muß in Apostrophen stehen und darf aus den Hexadezimalzeichen 0 – 9 und A – F bestehen. Ein Datenbyte wird jeweils durch zwei Hexadezimalzeichen dargestellt.

Die hexadezimale Darstellung eines Zeichens ist unterschiedlich, je nachdem ob der verwendete Computer ASCII- oder EBCDIC-Zeichensatz verwendet. Wenn Sie hexadezimale Konstanten auf einen anderen Computer übertragen, müssen Sie daher gegebenenfalls eine Zeichenumsetzung vornehmen.

ASCII-Beispiele:

H'313233' (entspricht der alphanumerischen Konstanten '123')

H'414243' (entspricht der alphanumerischen Konstanten 'ABC')

EBCDIC-Beispiele:

H'F1F2F3' (entspricht der alphanumerischen Konstanten '123')

H'C1C2C3' (entspricht der alphanumerischen Konstanten 'ABC')

Hexadezimale Konstanten können durch einen Bindestrich miteinander verkettet werden.

ASCII-Beispiel:

H'414243' – H'444546' (entspricht 'ABCDEF')

EBCDIC-Beispiel:

H'C1C2C3' – H'C4C5C6' (entspricht 'ABCDEF')

Auf diese Weise können hexadezimale Konstanten auch mit alphanumerischen Konstanten verkettet werden.

Anmerkung:

Wenn eine hexadezimale Konstante in ein anderes Feld übertragen wird, dann wird sie als alphanumerischer Wert behandelt.

Wenn unter UNIX eine hexadezimale Konstante ausgegeben wird, die Zeichen aus den Bereichen H'00' bis H'1F' oder H'80' bis H'A0' enthält, werden diese Zeichen nicht mit ausgegeben, da sie als Terminal-Steuerzeichen interpretiert würden. Mit Version 2.2 werden diese hexadezimalen Konstanten nicht mehr unterdrückt.

Logische Konstanten

Die logischen Konstanten “TRUE” (wahr) und “FALSE” (falsch) können einer Variablen zugeordnet werden, die mit Format L definiert ist.

Beispiel:

```
DEFINE DATA LOCAL
1 #FLAG (L)
END-DEFINE
...
MOVE TRUE TO #FLAG
...
IF #FLAG ...
    statement ...
    MOVE FALSE TO #FLAG
END-IF
...
```

Gleitkomma-Konstanten

Gleitkomma-Konstanten können in Verbindung mit Variablen, die das Format F haben, verwendet werden.

Beispiel:

```
DEFINE DATA LOCAL
1 #FLT1 (F4)
END-DEFINE
...
COMPUTE #FLT1 = -5.34E+2
...
```

Informationen zur Arithmetik mit Gleitkomma-Zahlen finden Sie auf Seite 81.

Attribut-Konstanten

Attribut-Konstanten können in Verbindung mit Variablen, die das Format C haben, benutzt werden. Attribut-Konstanten müssen in Klammern stehen.

Die folgenden Attribute können verwendet werden:

AD=D	default, standard	CD=BL	blau
AD=B	blinkend	CD=GR	grün
AD=I	intensiviert	CD=NE	neutral
AD=N	nicht angezeigt	CD=PI	pink
AD=V	invers	CD=RE	red, rot
AD=U	unterstrichen	CD=TU	türkis
AD=C	kursiv	CD=YE	yellow, gelb
AD=Y	dynamisches Attribut		
AD=P	protected, geschützt		

Beispiel:

```

DEFINE DATA LOCAL
1 #ATTR (C)
1 #FIELD (A10)
END-DEFINE
...
MOVE (AD=I CD=BL) TO #ATTR
...
INPUT #FIELD (CV=#ATTR)
...

```

Handle-Konstanten

Die Handle-Konstante NULL-HANDLE kann in Verbindung mit “GUI-Handles” und “Objekt-Handles” (object handles) verwendet werden.

Nähere Informationen zu “GUI-Handles” siehe *Natural User's Guide for Windows*.

Nähere Informationen zu “Objekt-Handles” siehe *NaturalX Documentation*.

Report-Spezifikation (*rep*)

Falls ein Programm mehrere Ausgabe-Reports erzeugt, muß jedes ausgabe-erzeugende Statement, das sich *nicht* auf den ersten auszugebenden Report (Report 0) bezieht, eine Report-Spezifikation erhalten. Falls nichts anderes angegeben wird, bezieht sich ein Statement auf den ersten ausgegebenen Report (Report 0).

Mit der Notation “(*rep*)” kann ein bestimmter anderer Report angegeben werden, auf den sich ein Statement beziehen soll. Es kann ein Wert von 1 bis 31 angegeben werden.

Auf Großrechnern gilt diese Notation nur für Batch-Reports, Reports unter Complete, CMS IMS/TM und TIAM, oder beim Einsatz von Natural Advanced Facilities unter CICS, TSO oder UTM.

Beispiele:

```
DISPLAY (1) NAME ...
```

```
WRITE (4) NAME ...
```

Statt eines Wertes von 1 bis 31 kann auch ein logischer Name, der mit einem DEFINE PRINTER-Statement zugewiesen wurde, angegeben werden.

Beispiel:

```
DEFINE PRINTER (LIST=5) OUTPUT 'LPT1'  
WRITE (LIST) NAME ...
```

Text-Notation

Mit einem der Statements INPUT, DISPLAY, WRITE, WRITE TITLE oder WRITE TRAILER ausgegebener Text muß entweder in Apostrophen (') oder in Anführungszeichen (") stehen (sogenannte *'text'*-Notation). Der *text* darf 1 bis 72 Zeichen lang sein und darf nicht über das Ende einer Sourcecode-Zeile hinausgehen. Zwei Textelemente können mittels eines Bindestriches verkettet werden.

Beispiele:

```
REINPUT 'PLEASE ENTER A VALID VALUE'  
  
WRITE    "NEW SALARY" #NEW-SALARY  
  
WRITE    'TEXT1'-'TEXT2'-'TEXT3'
```

Für ein Apostroph, das Teil eines in Apostrophen stehenden *text*-Elements ist, schreiben Sie entweder doppelte Apostrophe oder ein einzelnes Anführungszeichen; beides wird dann bei der Ausgabe in ein einzelnes Apostroph umgesetzt. Für ein Apostroph, das Teil eines in Anführungszeichen stehenden *text*-Elements ist, schreiben sie ein einzelnes Apostroph.

Beispiele:

```
#FIELDA = 'O' 'CONNOR'  
  
#FIELDA = 'O"CONNOR'  
  
#FIELDA = "O'CONNOR"
```

In allen drei Fällen erhalten Sie:

```
O'CONNOR
```

Anmerkung:

Falls Anführungszeichen nicht wie oben gezeigt in Apostrophe umgesetzt werden, liegt dies an der Einstellung des Profilparameters TQ; Näheres hierzu erfahren Sie von Ihrem Natural-Administrator.

Soll als Text ein einzelnes Zeichen mehrmals wiederholt werden, verwenden Sie dazu folgende Notation:

' c '(n)

c steht hierbei für das auszugebende Zeichen, und mit n geben Sie an, wie oft das Zeichen generiert werden soll. n darf maximal 249 betragen.

Beispiel:

WRITE '* '(3)

Statt der Apostrophe vor und nach dem Zeichen c können Sie auch Anführungszeichen verwenden.

Kommentare

Sie haben folgende Möglichkeiten, im Sourcecode Kommentare einzufügen:

- Falls Sie eine ganze Sourcecode-Zeile als Kommentarzeile verwenden möchten, geben Sie am Anfang der Zeile folgendes ein:
 - einen Stern und ein Leerzeichen (`*`)
 - zwei Sterne (`**`) oder
 - einen Schrägstrich und einen Stern (`/*`):

```
* USER COMMENT  
** USER COMMENT  
/* USER COMMENT
```

- Falls Sie nur einen Teil einer Sourcecode-Zeile für einen Kommentar verwenden möchten, geben Sie ein Leerzeichen, einen Schrägstrich und einen Stern ein (`/*`); der Rest der Zeile ab dieser Markierung ist damit als Kommentar gekennzeichnet:

```
ADD 5 TO #A      /* USER COMMENT
```

Ende eines Statements

Um das Ende eines Statements ausdrücklich zu kennzeichnen, können Sie zwischen dem Statement und dem nächsten Statement ein Semikolon (;) setzen. Dies kann zur Verdeutlichung der Programmstruktur verwendet werden, ist aber nicht erforderlich.

Logische Bedingungen

Folgende logische Bedingungen stehen zur Verfügung:

- relationaler Ausdruck (Seite 47)
- erweiterter relationaler Ausdruck (Seite 51)
- MASK-Option (Seite 52)
- SCAN-Option (Seite 61)
- BREAK-Option (Seite 63)
- IS-Option (Seite 65)
- Auswertung einer logischen Variablen (Seite 67)
- modifizierte Kontrollvariablen (Seite 69)
- SPECIFIED-Option (Seite 71)
- Felder in logischen Bedingungen (Seite 72)
- Logische Operatoren in komplexen logischen Ausdrücken (Seite 74).

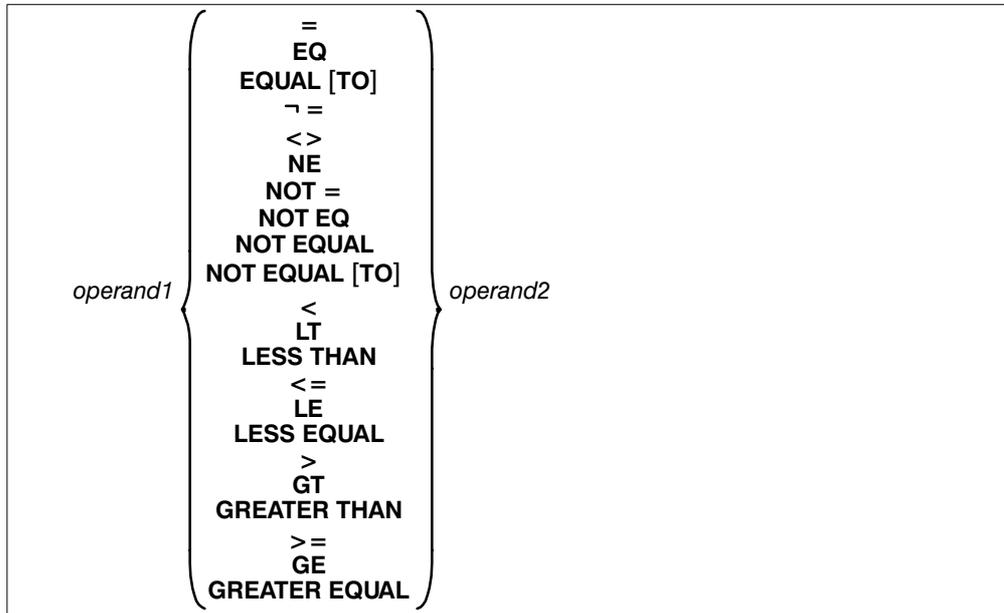
Die Grundform einer logischen Bedingung ist ein relationaler (vergleichender) Ausdruck. Mit den logischen Operatoren AND und OR können mehrere relationale Ausdrücke zu komplexen logischen Bedingungen verknüpft werden.

Arithmetische Ausdrücke können ebenfalls in logischen Bedingungen verwendet werden.

Logische Bedingungen können in den folgenden Statements angegeben werden:

Statement	Bedingungen
FIND	Zusätzlich zu dem primären Selektionskriterium, das in der WITH-Klausel angegeben wird, kann in einer WHERE-Klausel als zusätzliches Selektionskriterium eine logische Bedingung angegeben werden. Die in der WHERE-Klausel angegebene Bedingung wird erst ausgewertet, <i>nachdem</i> ein Datensatz aufgrund des WITH-Kriteriums ausgewählt und gelesen worden ist. In der WITH-Klausel werden “basic search criteria” (Suchkriterien) angegeben (vgl. FIND-Statement), aber keine logische Bedingung.
READ	In einer WHERE-Klausel kann eine logische Bedingung angegeben werden, die darüber entscheidet, ob ein gerade gelesener Datensatz weiterverarbeitet wird oder nicht. Diese Bedingung wird erst ausgewertet, <i>nachdem</i> ein Datensatz gelesen wurde.
HISTOGRAM	In einer WHERE-Klausel kann eine logische Bedingung angegeben werden, die darüber entscheidet, ob ein gerade gelesener Datensatz weiterverarbeitet wird oder nicht. Diese Bedingung wird erst ausgewertet, <i>nachdem</i> ein Datensatz gelesen wurde.
ACCEPT/REJECT	Zusätzlich zu den Selektionskriterien, aufgrund deren ein Datensatz mit einem FIND-, READ- oder HISTOGRAM-Statement gelesen wurde, kann in der IF-Klausel eines ACCEPT- oder REJECT-Statements eine weitere logische Bedingung angegeben werden, die über die weitere Verarbeitung eines Datensatzes entscheidet. Diese Bedingung wird erst ausgewertet, <i>nachdem</i> ein Datensatz gelesen wurde und die Verarbeitung des Datensatzes begonnen hat.
IF	Die Ausführung des Statements kann von der Erfüllung einer logischen Bedingung abhängig gemacht werden.
DECIDE FOR	Die Ausführung des Statements kann von der Erfüllung einer logischen Bedingung abhängig gemacht werden.
REPEAT	In der UNTIL- oder WHILE-Klausel eines REPEAT-Statements kann eine logische Bedingung angegeben werden, die darüber entscheidet, wann eine Verarbeitungsschleife beendet werden soll.

Relationaler Ausdruck



Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A N E	A N P I F B D T L G O	ja	ja
Operand2	C S A N E	A N P I F B D T L G O	ja	nein

Obige Operandentabelle ist im *Natural Statements-Handbuch* unter **Syntax-Symbole und Operandentabellen** erklärt. In der Spalte “Mögliche Struktur” der Tabelle steht “E” für “arithmetic expression” (arithmetischer Ausdruck), d.h. innerhalb eines relationalen Ausdrucks kann ein beliebiger arithmetischer Ausdruck als Operand verwendet werden.

Beispiele:

```
IF NAME = 'SMITH'
```

```
IF LEAVE-DUE GT 40
```

```
IF NAME = #NAME
```

Weitere Informationen über den Vergleich von Arrays in einem relationalen Ausdruck, siehe **Verarbeitung von Arrays** (Seite 89).

Anmerkung:

Wird ein Gleitkomma-Operand verwendet, so erfolgt der Vergleich in Gleitkommaform. Da Gleitkomma-Zahlen per se nur eine begrenzte Genauigkeit haben, lassen sich Rundungs- bzw. Abschneidefehler bei der Konvertierung von Zahlen in/aus Gleitkommaform nicht ausschließen (vgl. Seite 81).

Arithmetische Ausdrücke in logischen Bedingungen

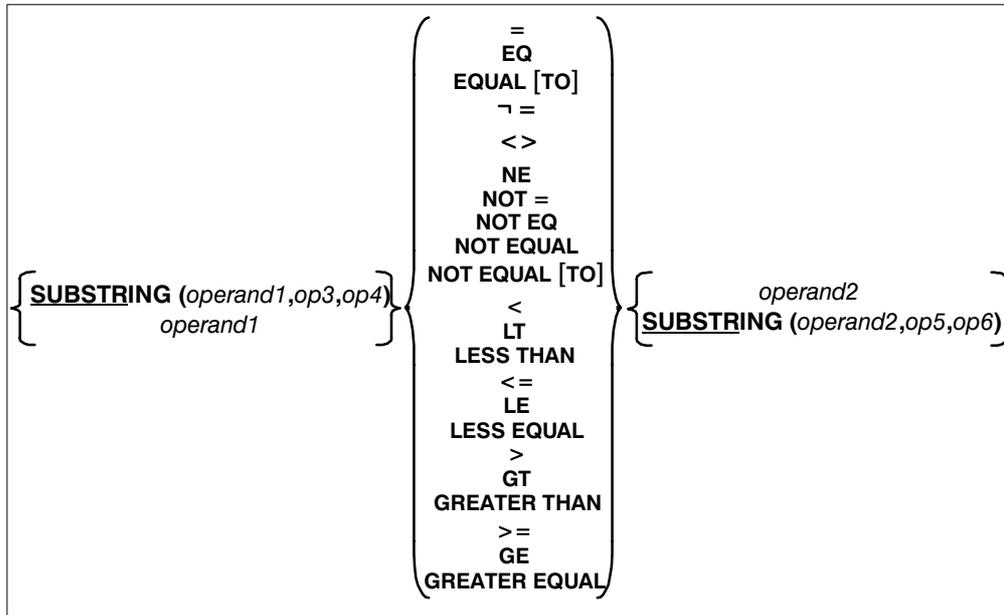
Das folgende Beispiel zeigt, wie arithmetische Ausdrücke in logischen Bedingungen eingesetzt werden können:

```
IF #A + 3 GT #B - 5 AND #C * 3 LE #A + #B
```

Handles in logischen Bedingungen

Wenn die Operanden in einem relationalen Ausdruck Handles sind, dürfen nur EQUAL- und NOT EQUAL-Operatoren verwendet werden.

SUBSTRING-Option in relationalem Ausdruck



Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A N	A	ja	ja
Operand2	C S A N	A	ja	nein
Op3 / Op5	C S	N P I	ja	nein
Op4 / Op6	C S	N P I	ja	nein

Mit der SUBSTRING-Option können Sie einen *Teil* eines alphanumerischen Feldes vergleichen. Nach dem Feldnamen (*operand1*) geben Sie zunächst die erste Stelle (*op3*) und dann die Länge (*op4*) des zu vergleichenden Feldteils an.

Sie können auch einen Feldwert mit einem *Teil* eines anderen Feldwertes vergleichen. Nach dem Feldnamen (*operand2*) geben Sie zunächst die erste Stelle (*op5*) und dann die Länge (*op6*) des Feldteils an, mit dem *operand1* verglichen werden soll.

Sie können auch beide Formen miteinander kombinieren, d.h. SUBSTRING gleichzeitig für *operand1* und für *operand2* angeben.

Beispiele:

Dieser Ausdruck vergleicht die 5. bis einschließlich 12. Stelle des Wertes in Feld #A mit dem Wert von Feld #B:

`SUBSTRING(#A,5,8) = #B`

Dieser Ausdruck vergleicht den Wert von Feld #A mit der 3. bis einschließlich 6. Stelle des Wertes in Feld #B:

`#A = SUBSTRING(#B,3,4)`

Anmerkung:

Wenn Sie op3/op5 weglassen, wird ab Anfang des Feldes verglichen. Wenn Sie op4/op6 weglassen, wird ab der angegebenen Stelle (op3/op5) bis zum Ende des Feldes verglichen.

Erweiterter Relationaler Ausdruck

$$\begin{array}{c}
 \text{operand1} \left\{ \begin{array}{c} \text{EQ} \\ \text{EQUAL [TO]} \end{array} \right\} \text{operand2} \\
 \\
 \left\{ \begin{array}{c} \text{OR} \left\{ \begin{array}{c} \text{EQ} \\ \text{EQUAL [TO]} \end{array} \right\} \text{operand3} \\
 \dots \\
 \text{THRU operand4 [BUT NOT operand5 [THRU operand 6]]} \end{array} \right\}
 \end{array}$$

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A N*E	A N P I F B D T G O	ja	nein
Operand2	C S A N*E	A N P I F B D T G O	ja	nein
Operand3	C S A N*E	A N P I F B D T G O	ja	nein
Operand4	C S A N*E	A N P I F B D T G O	ja	nein
Operand5	C S A N*E	A N P I F B D T G O	ja	nein
Operand6	C S A N*E	A N P I F B D T G O	ja	nein

* Mathematische Funktionen und Systemvariablen sind erlaubt.
Gruppenwechsel-Funktionen sind nicht erlaubt.

Operand3 kann auch unter Verwendung einer MASK- oder SCAN-Option angegeben werden; d.h. er kann angegeben werden als:

MASK (*mask-definition*) [*operand*]

MASK *operand*

SCAN *operand*

Einzelheiten zu diesen Optionen finden Sie unter **MASK-Option** (Seite 52) und **SCAN-Option** (Seite 61).

Beispiele:

IF #A = 2 OR = 4 OR = 7

IF #A = 5 THRU 11 BUT NOT 7 THRU 8

MASK-Option

Mit der MASK-Option können Sie bestimmte ausgewählte Stellen eines Feldes nach einem bestimmten Wert absuchen.

Konstante Maske

$$\text{operand1} \left\{ \begin{array}{c} = \\ \text{EQ} \\ \text{EQUAL TO} \\ \text{NE} \\ \text{NOT EQUAL} \end{array} \right\} \text{MASK (mask-definition) [operand2]}$$

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A N	A N P	ja	nein
Operand2	C S	A N P B	ja	nein

Operand2 kann nur angegeben werden, wenn die *mask-definition* mindestens ein "X" enthält. *Operand1* und *operand2* müssen format-kompatibel sein: wenn *operand1* Format A hat, muß *operand2* Format A, B oder N haben; wenn *operand1* Format N oder P hat, muß *operand2* Format N oder P haben. Wird ein "X" in der *mask-definition* angegeben, werden die betreffenden inhaltlichen Stellen von *operand1* und *operand2* zum Wertevergleich ausgewählt.

Variable Maske

Anstatt einer konstanten *mask-definition* (siehe oben) können Sie auch eine variable MASK-Definition angeben:

$$\text{operand1} \left\{ \begin{array}{c} = \\ \text{EQ} \\ \text{EQUAL TO} \\ \text{NE} \\ \text{NOT EQUAL} \end{array} \right\} \text{MASK operand2}$$

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A N	A N P	ja	nein
Operand2	S	A	ja	nein

Der Inhalt von *operand2* wird dann als MASK-Definition genommen. Nachgestellte Leerzeichen in *operand2* werden ignoriert.

Zeichen in einer Maske

In einer *mask-definition* können Sie folgende Zeichen verwenden (die Masken-Definition ist bei einer konstanten Maske in der *mask-definition* und bei einer variablen Maske in *operand2* enthalten):

Zeichen	Bedeutung
. oder ? oder _	Eine einzelne Stelle, die nicht überprüft werden soll.
* oder %	Eine beliebige Anzahl von Stellen, die nicht überprüft werden sollen.
/	(Schrägstrich) Prüft, ob der Wert mit einem (oder mehreren) bestimmten Zeichen endet. Beispiel: Die folgende Bedingung ist wahr, wenn entweder an der letzten Stelle des Feldes ein "E" steht oder nach dem "E" nur noch Leerzeichen stehen: <code>IF #FIELD = MASK (*'E' /)</code>
A	Eine Stelle, die nach Groß- oder Kleinbuchstaben abgesucht werden soll.
'c'	Eine oder mehrere Stellen, die nach den in Apostrophen stehenden Zeichen abgesucht werden sollen (doppelte Apostrophe bedeuten, daß innerhalb der Zeichenkette nach einem Apostrophen gesucht wird).
C	Eine Stelle, die nach alphanumerischem Inhalt (Groß- oder Kleinbuchstabe, Zahl) oder Leerzeichen abgesucht werden soll.
DD	Zwei Stellen, die nach einem gültigen Tagesdatum abgesucht werden sollen (01 – 31; abhängig von den Werten für MM und YY/YYYY, falls angegeben; vgl. Datumsprüfungen , Seite 55).
H	Eine Stelle, die nach einem Hexadezimalzeichen (A – F, 0 – 9) abgesucht werden soll.
L	Eine Stelle, die nach Kleinbuchstaben (a – z) abgesucht werden soll.
MM	Zwei Stellen, die nach einer gültigen Monatsangabe (01 – 12) abgesucht werden sollen.
N	Eine Stelle, die nach einer Ziffer abgesucht werden soll.
n...	Eine oder mehrere Stellen, die nach einem numerischen Wert im Bereich von 0 bis <i>n</i> abgesucht werden sollen.
<i>n1–n2</i> oder <i>n1:n2</i>	Stellen, die nach einem numerischen Wert im Bereich von <i>n1–n2</i> abgesucht werden sollen. <i>n1</i> und <i>n2</i> müssen gleich lang sein.
P	Eine Stelle, die nach einem druckbaren Zeichen (U, L, N oder S – Buchstabe, Zahl oder Sonderzeichen) abgesucht werden soll.
S	Eine Stelle, die nach Sonderzeichen abgesucht werden soll.

Zeichen	Bedeutung
U	Eine Stelle, die nach Großbuchstaben (A – Z) abgesucht werden soll.
X	Eine Stelle, die mit der entsprechenden Stelle des auf die <i>mask-definition</i> folgenden Wertes (<i>operand2</i>) verglichen werden soll. In einer variablen Maske ist “X” nicht erlaubt, da sinnlos.
YY	Zwei Stellen, die nach einer gültigen Jahreszahl (00 – 99) abgesucht werden sollen. Siehe auch Datumsprüfungen (Seite 55).
YYYY	Vier Stellen, die nach einer gültigen Jahreszahl (0000 – 2699) abgesucht werden sollen.
Z	Eine Stelle, die nach einem Zeichen abgesucht werden soll, dessen linkes Halbbyte hexadezimal 3 oder 7 (ASCII) bzw. A – F (EBCDIC) und dessen rechtes Halbbyte hexadezimal 0 – 9 ist. Damit können Sie korrekt nach Ziffern in negativen Zahlen suchen. Mit “N” (womit Sie eine Stelle nach einer Ziffer absuchen können) erhalten Sie bei Suche von Ziffern in negativen Zahlen falsche Ergebnisse, da das Vorzeichen in der letzten Stelle der Zahl gespeichert ist, wodurch diese Stelle hexadezimal als Nicht-Ziffer dargestellt wird. Geben Sie innerhalb einer Maske für jede Reihe numerischer Stellen, die geprüft werden sollen, nur jeweils ein “Z” an.

Maskenlänge

Welche Stellen abgesucht werden sollen, ergibt sich aus der Definition der Maske.

Beispiel:

```
DEFINE DATA LOCAL
1 #CODE (A15)
END-DEFINE
...
IF #CODE = MASK (NN'ABC'....NN)
...
```

Die ersten beiden Stellen von #CODE werden nach einem numerischen Wert abgesucht, die 3. bis 5. Stelle nach dem Wert "ABC", die 10. und 11. Stelle nach einem numerischen Wert; die 6. bis 9. und 12. bis 15. Stelle werden nicht überprüft.

Datumsprüfungen

Pro Maske darf nur ein Datum geprüft werden.

Wird bei der Prüfung eines Tagesdatums (DD) keine Monatsangabe (MM) in der Maske gemacht, wird der aktuelle Monat angenommen.

Wird bei der Prüfung eines Tagesdatums (DD) keine Jahresangabe (YY bzw. YYYY) in der Maske gemacht, wird das aktuelle Jahr angenommen.

Bei der Prüfung einer zweistelligen Jahreszahl (YY) wird das aktuelle Jahrhundert angenommen, sofern der Profilparameter YSLW auf "0" gesetzt ist. Ist der YSLW-Parameter auf einen anderen Wert gesetzt, wird das Jahrhundert entsprechend des "Year Sliding Window" berechnet (wie unter Profilparameter YSLW in diesem Handbuch und in der *Natural Operations Documentation* beschrieben).

Beispiele:

Im folgenden Beispiel werden Monat und Tag auf ihre Gültigkeit überprüft. Der Monatswert "11" ist gültig, wohingegen der Tageswert "31" ungültig ist, da der 11. Monat nur 30 Tage hat.

Beispiel 1:

```
MOVE 1131 TO #DATE (N4)  
IF #DATE = MASK (MMDD)
```

Beispiel 2:

Im folgenden Beispiel wird überprüft, ob das Feld #DATE ein gültiges Datum im Format MM/DD/YY (Monat/Tag/Jahr) enthält.

```
IF #DATE(A8) = MASK (MM'/'DD'/'YY)
```

Beispiel 3:

```
IF #DATE(A4) = MASK (19-20YY)
```

Im folgenden Beispiel wird überprüft, ob das Feld #DATE eine zweistellige Zahl im Bereich von 19 bis 20, gefolgt von einem gültigen zweistelligen Jahr (00 bis 99) enthält. Das Jahrhundert wird wie oben beschrieben von Natural angegeben.

Anmerkung: Obwohl dies offensichtlich ist, ermöglicht die oben angegebene Maske nicht das Abprüfen auf ein gültiges Jahr im Bereich von 1900 bis 2099, weil der numerische Wertebereich 19-20 unabhängig von der Gültigkeitsprüfung für das Jahr abgeprüft wird.

Um auf Bereiche von Jahren abzuprüfen, bauen Sie eine Gültigkeitsprüfung für das Datum und eine andere für den Bereich in Ihrem Programm ein:

```
IF #DATE(A10) = MASK (YYYY'-'MM'-'DD) AND #DATE = MASK (19-20)
```

Prüfung unter Verwendung von Konstanten oder Variablen

Ist der für die Maskenprüfung verwendete Wert eine Konstante oder der Inhalt einer Variablen, dann muß dieser Wert (*operand2*) unmittelbar nach der *mask-definition* angegeben werden.

Operand2 muß mindestens so lang sein wie die Maske.

In der Maske geben Sie für jede zu überprüfende Stelle ein “X” und für jede nicht zu überprüfende Stelle “.” (oder “?” oder “_”) an.

Beispiel:

```
DEFINE DATA LOCAL
1 #NAME (A15)
END-DEFINE
...
IF #NAME = MASK (. . XX) 'ABCD'
...
```

Hier wird geprüft, ob die 3. bis 4. Stelle des Feldes #NAME den Wert “CD” enthält. Die ersten beiden Stellen werden nicht überprüft.

Wieviele Stellen geprüft werden, hängt von der Länge der definierten Maske ab. Die Maske wird immer linksbündig auf das zu überprüfende Feld bzw. die zu prüfende Konstante ausgerichtet. *Operand1* muß dasselbe Format haben wie *operand2*.

Hat das zu überprüfende Feld (*operand1*) Format A, muß ein konstanter Wert (*operand2*) in Apostrophen stehen. Ist das Feld numerisch, muß der Wert eine Zahl oder der Inhalt eines numerischen Datenbankfeldes bzw. einer numerischen Benutzervariablen sein.

In jedem Fall werden Zeichen/Stellen, die nicht auf einer in der Maskendefinition mit “X” markierten Stelle liegen, ignoriert.

Die MASK-Bedingung ist erfüllt, wenn alle in der Maske angegebenen Stellen dem geforderten Wert entsprechen.

Beispiel:

```
/* EXAMPLE 'LCCMASK'
/* EXAMPLE OF USING MASK OPTION WITHIN LOGICAL CONDITION
/*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 CITY
END-DEFINE
/*****
HISTOGRAM EMPLOY-VIEW CITY
  IF CITY = MASK (...XX) '...NN'
    DISPLAY NOTITLE CITY *NUMBER
  END-IF
END-HISTOGRAM
/*****
END
```

In diesem Beispielprogramm werden nur Datensätze akzeptiert, bei denen das Feld CITY einen Wert enthält, der an der 5. und 6. Stelle jeweils ein "N" hat.

Bereichsprüfungen

Bei Bereichsprüfungen wird die Anzahl der verifizierten Stellen durch die Genauigkeit des in der Maske angegebenen Wertes definiert. Zum Beispiel würde die Maske “(...193...)” die Stellen 4 bis 6 nach einer dreistelligen Zahl im Bereich von 000 bis 193 überprüfen.

Weitere Beispiele für Masken-Definitionen:

- In diesem Beispiel wird überprüft, ob alle Stellen des Feldes #NAME einen Buchstaben enthalten:

```
IF #NAME (A10) = MASK (AAAAAAAAA)
```

- In diesem Beispiel wird überprüft, ob die 4.–6. Stelle von #NUMBER eine Zahl enthält:

```
IF #NUMBER (A6) = MASK (...NNN)
```

- In diesem Beispiel wird überprüft, ob die 4.–6. Stelle von #VALUE den Wert “123” enthält:

```
IF #VALUE(A10) = MASK (... '123')
```

- In diesem Beispiel wird überprüft, ob das Nummernschild-Feld #LICENSE ein KFZ-Kennzeichen enthält, das mit “NY–” beginnt, gefolgt vom Wert der letzten fünf Stellen des Feldes #VALUE:

```
DEFINE DATA LOCAL
  1 #VALUE (A8)
  1 #LICENSE (A8)
END-DEFINE
INPUT 'ENTER KNOWN POSITIONS OF LICENSE PLATE:' #VALUE
IF #LICENSE = MASK ('NY-'XXXXX) #VALUE
```

- Die folgende Bedingung wird von jedem Wert erfüllt, der “NAT” und “AL” enthält, ganz gleich wieviele andere Zeichen zwischen “NAT” und “AL” stehen (dies würde z.B. auf die Werte NATURAL und NATIONALITAET genauso zutreffen wie auf den Wert NATAL):

```
MASK ('NAT'*'AL')
```

Auf gepackte oder ungepackte numerische Daten abprüfen

In Altanwendungen sind gepackte oder ungepackte numerische Variablen häufig mit alphanumerischen oder binären Feldern neu definiert. Solche Neudefinitionen sind nicht empfehlenswert, weil die Verwendung einer gepackten oder ungepackten Variablen in einer Zuweisung oder Berechnung zu Fehlern oder unvorhersagbaren Ergebnissen führen kann.

Um den Inhalt einer solchen neu definierten Variablen auf Gültigkeit hin abzuprüfen, bevor die Variable verwendet wird, benutzen Sie die Option N so oft wie die Anzahl der Stellen – 1 mal, gefolgt von einer einzelnen Option Z.

Beispiele:

```
IF #P1 (P1) = MASK (Z)  
IF #N4 (N4) = MASK (NNNZ)  
IF #P5 (P5) = MASK (NNNNZ)
```

SCAN-Option

$$\text{operand1} \left\{ \begin{array}{c} = \\ \text{EQ} \\ \text{EQUAL TO} \\ \text{NE} \\ \text{NOT EQUAL} \end{array} \right\} \text{SCAN operand2}$$

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A N	A N P	ja	nein
Operand2	C S	A B*	ja	nein

* *Operand2* darf nur binär sein, wenn *operand1* alphanumerisch ist.

Mit der SCAN-Option können Sie nach einem bestimmten Wert in einem Feld suchen.

Der zu suchende Wert (*operand2*) kann entweder als alphanumerische Konstante (eine in Apostrophen stehende Zeichenkette) oder als Inhalt einer alphanumerischen Variablen (Datenbankfeld oder Benutzervariable) angegeben werden.

Dem Wert nachgestellte Leerzeichen werden automatisch eliminiert. Deshalb kann die SCAN-Option nicht zum Suchen nach Leerzeichen verwendet werden.

Falls *operand1* alphanumerisch ist, darf *operand2* auch binäres Format haben.

Das Feld, das abgesucht werden soll (*operand1*), kann das Format A, N oder P haben. Die SCAN-Operation kann mit den Operatoren EQ ("gleich") oder NE ("ungleich") angegeben werden.

Die Länge der gesuchten Zeichenkette sollte kürzer als die Länge des abgesuchten Feldes sein. Ist die Länge des Wertes gleich der des Feldes, sollte statt der SCAN-Option ein relationaler Ausdruck mit dem Operator EQUAL TO verwendet werden.

Beispiel für SCAN-Option:

```

/* EXAMPLE 'LCCSCAN'
/* EXAMPLE OF USING SCAN OPTION IN LOGICAL CONDITION
/*****
DEFINE DATA
  LOCAL
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 NAME
  1 #VALUE (A4)
  1 #COMMENT (A10) INIT <' '>
END-DEFINE
/*****
INPUT 'ENTER SCAN VALUE:' #VALUE
LIMIT 14
HISTOGRAM EMPLOY-VIEW NAME
  RESET #COMMENT
  IF NAME = SCAN #VALUE
    MOVE 'MATCH' TO #COMMENT
  END-IF
DISPLAY NOTITLE NAME *NUMBER #COMMENT
END-HISTOGRAM
/*****
END

```

```
ENTER SCAN VALUE: LL
```

NAME	NMBR	#COMMENT
ABELLAN	1	MATCH
ACHIESON	1	
ADAM	1	
ADKINSON	8	
AECKERLE	1	
AFANASSIEV	2	
AHL	1	
AKROYD	1	
ALEMAN	1	
ALESTIA	1	
ALEXANDER	5	
ALLEGRE	1	MATCH
ALLSOP	1	MATCH
ALTINOK	1	

BREAK (Gruppenwechsel) in einer logischen Bedingung

BREAK [OF] operand1 [InI]

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	S	A N P	ja	nein

Anmerkung:

Dynamische oder große Variablen dürfen nicht als operand1 verwendet werden.

Mit der BREAK-Option kann der aktuelle Wert eines Feldes (oder eines Teils eines Feldes) mit dem Wert verglichen werden, den das Feld im vorangegangenen Durchlauf durch die Verarbeitungsschleife hatte.

Mit *operand1* geben Sie das Feld an, das überprüft werden soll.

Soll nur ein Teil des Feldes überprüft werden, so geben Sie eine mit Schrägstrichen eingegrenzte Zahl *n* an, die angibt, wieviele Stellen (von links nach rechts gezählt) des Feldes auf einen Wertwechsel überprüft werden sollen.

Eine BREAK-Bedingung ist erfüllt, wenn der Wert des Kontrollfeldes (bzw. der angegebenen Stellen des Feldes) sich ändert. Eine BREAK-Bedingung ist nicht erfüllt, wenn eine AT END OF DATA-Bedingung auftritt.

Beispiel:

```
BREAK FIRST-NAME /1/
```

In diesem Beispiel wird überprüft, ob der Wert der ersten Stelle des Feldes FIRST-NAME sich geändert hat.

Der Einsatz von Natural-Systemfunktionen (die bei einem AT BREAK-Statement zur Verfügung stehen) ist bei der BREAK-Option *nicht* erlaubt.

Beispiel für BREAK-Option:

```

* EXAMPLE 'LCCBRK': BREAK OPTION IN LOGICAL CONDITION
*****
DEFINE DATA LOCAL
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 NAME
    2 FIRST-NAME
    2 BIRTH
  1 #BIRTH (A8)
END-DEFINE
*
LIMIT 10
READ EMPLOY-VIEW BY BIRTH
  MOVE EDITED BIRTH (EM=YYYYMMDD) TO #BIRTH
  IF BREAK OF #BIRTH /6/
    NEWPAGE IF LESS THAN 5 LINES LEFT
    WRITE / '-' (50) /
  END-IF
  DISPLAY NOTITLE BIRTH (EM=YYYY-MM-DD) NAME FIRST-NAME
END-READ
END

```

DATE OF BIRTH	NAME	FIRST-NAME
1940-01-01	GARRET	WILLIAM
1940-01-09	TAILOR	ROBERT
1940-01-09	PIETSCH	VENUS
1940-01-31	LYTTLETON	BETTY
1940-02-02	WINTRICH	MARIA
1940-02-13	KUNEY	MARY
1940-02-14	KOLENCE	MARSHA
1940-02-24	DILWORTH	TOM
1940-03-03	DEKKER	SYLVIA
1940-03-06	STEFFERUD	BILL

IS-Option — Format-/Längenprüfung eines Wertes

operand1 IS (*format*)

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	S A	A	ja	nein

Mit dieser Option können Sie prüfen, ob der Inhalt eines alphanumerischen Feldes (*operand1*) in ein bestimmtes anderes *Format* übertragbar ist.

Dieses *Format* kann sein:

Nll.ll	Numerisch mit Länge ll.ll.
Fll	Gleitkomma (floating point) mit Länge ll.
D	Datum. Folgende Datumsformate sind möglich: dd-mm-yy, dd-mm-yyyy, ddmmyyyy (dd = Tag, mm = Monat, yy bzw. yyyy = Jahr). Die Abfolge der Tages-, Monats- und Jahreskomponenten sowie die Trennzeichen zwischen den Komponenten werden durch den Profilparameter DTFORM (der in Ihrer <i>Natural Operations Documentation</i> beschrieben ist) bestimmt.
T	Zeit (time); entsprechend dem Standard-Zeitanzzeigeformat.
Pll.ll	Gepackt numerisch mit Länge ll.ll.
Ill	Ganzzahl (integer) mit Länge ll.

Bei der Prüfung werden für den Wert in *operand1* vor- oder nachgestellte Leerzeichen ignoriert.

Diese Prüfung ist sinnvoll, wenn beispielsweise vor Ausführung der mathematischen Funktion VAL (Erhalt des numerischen Wertes eines alphanumerischen Feldes) das Format des Wertes überprüft wird, um zu vermeiden, daß ein falsches Format einen Laufzeitfehler verursacht.

Anmerkung:

Mit der IS-Option kann nicht geprüft werden, ob der Wert eines alphanumerischen Feldes in dem angegebenen "format" ist, sondern ob er in das "format" übertragbar ist. Um zu prüfen, ob ein Wert in einem bestimmten Format ist, können Sie die MASK-Option (Seite 52) verwenden.

Beispiel für IS-Option:

```

/* EXAMPLE 'LCCFMT'
/* EXAMPLE OF FORMAT/LENGTH CHECK IN LOGICAL CONDITION
/*****
DEFINE DATA LOCAL
1 #FIELDA (A10)          /* INPUT FIELD TO BE CHECKED
1 #FIELDB (N5)          /* RECEIVING FIELD OF VAL FUNCTION
1 #DATE (A10)           /* INPUT FIELD FOR DATE
END-DEFINE
/*****
INPUT #DATE #FIELDA
IF #DATE IS (D)
  IF #FIELDA IS (N5)
    COMPUTE #FIELDB = VAL(#FIELDA)
    WRITE NOTITLE 'VAL FUNCTION OK' // '=' #FIELDA '=' #FIELDB
  ELSE
    REINPUT 'FIELD DOES NOT FIT INTO N5 FORMAT'
    MARK *#FIELDA
  END-IF
ELSE
  REINPUT 'INPUT IS NOT IN DATE FORMAT (YY-MM-DD) '
  MARK *#DATE
END-IF
/*****
END

```

```
#DATE 150487      #FIELDA
```

```
INPUT IS NOT IN DATE FORMAT (YY-MM-DD)
```

Auswertung einer logischen Variablen

<i>operand1</i>

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S A	L	nein	nein

Diese Option kann in Verbindung mit einer logischen Variablen (Format L) eingesetzt werden. Eine logische Variable kann die Werte "TRUE" (wahr) oder "FALSE" (falsch) haben. Mit *operand1* geben Sie den Namen der Variablen an.

Beispiel für logische Variable:

```

/* EXAMPLE 'LCCLLOG'
/* EXAMPLE OF LOGICAL VARIABLE IN LOGICAL CONDITION
/*****
DEFINE DATA LOCAL
1 #SWITCH (L) INIT <TRUE>
1 #INDEX (I1)
END-DEFINE
/*****
FOR #INDEX 1 5
  WRITE NOTITLE #SWITCH (EM=FALSE/TRUE) 5X 'INDEX =' #INDEX
  WRITE NOTITLE #SWITCH (EM=OFF/ON) 7X 'INDEX =' #INDEX
  IF #SWITCH
    MOVE FALSE TO #SWITCH
  ELSE
    MOVE TRUE TO #SWITCH
  END-IF
/*****
SKIP 1
END-FOR
END

```

TRUE	INDEX =	1
ON	INDEX =	1
FALSE	INDEX =	2
OFF	INDEX =	2
TRUE	INDEX =	3
ON	INDEX =	3
FALSE	INDEX =	4
OFF	INDEX =	4
TRUE	INDEX =	5
ON	INDEX =	5

Modifizierte Kontrollvariablen

operand1 [NOT] MODIFIED

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	S A	C	nein	nein

Mit dieser Option wird überprüft, ob der Inhalt eines Feldes, dessen Attribute dynamisch zugewiesen werden, während der Ausführung eines INPUT-Statements verändert worden ist.

Von einem INPUT-Statement referenzierte Kontrollvariablen erhalten immer den Status “NOT MODIFIED” (nicht verändert), wenn die Map am Terminal ausgegeben wird.

Wird der Inhalt eines Feldes, das eine Kontrollvariable (*operand1*) referenziert, verändert, erhält die Kontrollvariable den Status “MODIFIED” (verändert). Referenzieren mehrere multiple Felder dieselbe Kontrollvariable, so erhält die Variable den Status “MODIFIED”, wenn mindestens eines dieser Felder verändert wurde.

Ist die Kontrollvariable *operand1* ein Array, so erhält die Variable den Status “MODIFIED”, wenn mindestens eins der Elemente des Arrays verändert wurde (OR-Verknüpfung).

Anmerkung:

*Auf Großrechnern kann mit dem Profilparameter CVMIN (siehe Natural Operations for Mainframes Documentation) festgelegt werden, ob eine Kontrollvariable auch dann auf “MODIFIED” gesetzt werden soll, wenn der Wert des betreffenden Feldes durch einen **identischen** Wert überschrieben wurde.*

Beispiel für modifizierte Kontrollvariable:

```

/* EXAMPLE 'LCCMOD'
/* EXAMPLE OF MODIFIED FIELD CHECK IN LOGICAL CONDITION
/*****
DEFINE DATA LOCAL
1 #ATTR (C)
1 #A (A1)
1 #B (A1)
END-DEFINE
/*****
MOVE (AD=I) TO #ATTR
/*****
INPUT (CV=#ATTR) #A #B
IF #ATTR NOT MODIFIED
  WRITE NOTITLE 'FIELD #A OR #B HAS NOT BEEN MODIFIED'
END-IF
/*****
IF #ATTR MODIFIED
  WRITE NOTITLE 'FIELD #A OR #B HAS BEEN MODIFIED'
END-IF
/*****
END

```

```
#A x #B
```

```
FIELD #A OR #B HAS BEEN MODIFIED
```

SPECIFIED-Option

parameter-name [NOT] SPECIFIED

Diese Option steht auf Großrechnern nicht zur Verfügung.

Die SPECIFIED-Option wird verwendet, um zu überprüfen, ob ein optionaler Parameter in einem aufgerufenen Objekt (Subprogramm, externes Unterprogramm, Dialog oder ActiveX Control) von dem aufrufenden Objekt einen Wert erhalten hat oder nicht.

Ein optionaler Parameter ist ein mit dem Schlüsselwort OPTIONAL im DEFINE DATA PARAMETER-Statement des aufgerufenen Objekts definiertes Feld. Wenn ein Feld als OPTIONAL definiert ist, kann ein Wert als Option von einem aufgerufenen Objekt an dieses Feld übergeben werden.

Im aufgerufenen Statement wird die Notation *nX* verwendet, um Parameter anzugeben, für die keine Werte übergeben werden.

Wenn ein optionaler Parameter verarbeitet wird, an den kein Wert übergeben wurde, so führt dies zu einem Laufzeitfehler. Um solch einen Fehler zu verhindern, verwenden Sie in dem aufgerufenen Objekt die SPECIFIED-Option, um zu prüfen, ob an einen optionalen Parameter ein Wert übergeben wurde oder nicht. Er wird erst dann verarbeitet, wenn dies der Fall ist.

Parameter-name ist der Name des im DEFINE DATA PARAMETER-Statement des aufgerufenen Objekts angegebenen Parameters.

Bei einem nicht als OPTIONAL definierten Feld ist die SPECIFIED-Bedingung immer "TRUE" (wahr).

Felder in logischen Bedingungen

Bei der Konstruktion logischer Bedingungen dürfen sowohl Datenbankfelder als auch Benutzervariablen verwendet werden. Datenbankfelder, die Teil einer Periodengruppe oder multiple Felder sind, dürfen ebenfalls verwendet werden. Wenn ein Bereich von Werten für multiple Felder oder ein Bereich von Ausprägungen für Periodengruppen angegeben wird, dann ist die Bedingung erfüllt, wenn der Suchwert in einem/r Wert/Ausprägung innerhalb des angegebenen Bereichs gefunden wird.

Jeder verwendete Wert muß mit dem ihm in einem relationalen Ausdruck gegenüberstehenden Feld kompatibel sein. Dezimalstellen können nur bei Werten für numerische Felder angegeben werden, wobei die Anzahl der Dezimalstellen von Wert und Feld kompatibel sein muß.

Haben zwei Operanden unterschiedliches Format, wird das Format des zweiten Operanden in das des ersten umgesetzt.

Die folgende Tabelle zeigt, welche Operandenformate in einer logischen Bedingung zusammen verwendet werden können:

Operand2 → ↓ Operand1	A	Bn (n≤4)	Bn (n≥5)	D	T	I	F	L	N	P	GH	OH
A	✓	✓	✓									
Bn (n≤4)	✓	✓	✓			✓	✓		✓	✓		
Bn (n≥5)	✓	✓	✓									
D				✓								
T					✓							
I		✓				✓	✓		✓	✓		
F		✓				✓	✓		✓	✓		
L												
N		✓				✓	✓		✓	✓		
P		✓				✓	✓		✓	✓		
GH											✓	
OH												✓

GH = GUI Handle, OH = Object Handle.

Wird ein Array mit einem Skalarwert in Relation gesetzt, so wird jedes Element des Arrays mit dem Skalarwert verglichen; die Bedingung ist erfüllt, wenn *mindestens ein* Array-Element die Bedingung erfüllt (Oder-Verknüpfung).

Wird ein Array mit einem Array in Relation gesetzt, so wird jedes Element des einen Arrays mit dem entsprechenden Element des anderen Arrays verglichen; die Bedingung ist nur dann erfüllt, wenn *alle* Element-Vergleiche die Bedingung erfüllen (Und-Verknüpfung).

Vgl. **Verarbeitung von Arrays** (Seite 89).

Phonetische Deskriptoren (Adabas) dürfen in einer logischen Bedingung nicht verwendet werden.

Beispiele für logische Bedingungen:

```
FIND EMPLOYEES-VIEW WITH CITY = 'BOSTON' WHERE SEX = 'M'
```

```
READ EMPLOYEES-VIEW BY NAME WHERE SEX = 'M'
```

```
ACCEPT IF LEAVE-DUE GT 45
```

```
IF #A GT #B THEN COMPUTE #C = #A + #B
```

```
REPEAT UNTIL #X = 500
```

Logische Operatoren in komplexen logischen Ausdrücken

Mittels der Boole'schen Operatoren "AND", "OR" und "NOT" ist es möglich, logische Bedingungen miteinander zu verknüpfen. Mit Hilfe von Klammern können logische Bedingungen logisch zusammengefaßt werden.

Die Operatoren werden in der folgenden Reihenfolge ausgewertet:

- ① () Klammer-Rechnung
- ② NOT Negation
- ③ AND Und-Verknüpfung
- ④ OR Oder-Verknüpfung

Die folgenden logischen Bedingungen (*logical-condition-criteria*) können mit logischen Operatoren verknüpft werden, um einen komplexen logischen Ausdruck (*logical-expression*) zu bilden: relationale Ausdrücke, erweiterte relationale Ausdrücke sowie MASK-, SCAN- und BREAK-Optionen.

Die Syntax für eine *logical-expression* ist wie folgt:

$$[\text{NOT}] \left\{ \begin{array}{l} \text{logical-condition-criterion} \\ \text{(logical-expression)} \end{array} \right\} \left[\left[\begin{array}{l} \text{OR} \\ \text{AND} \end{array} \right] \text{logical-expression} \right] \dots$$

Beispiele für "logical-expressions":

```
FIND STAFF-VIEW WITH CITY = 'TOKYO'
      WHERE BIRTH GT 19610101 AND SEX = 'F'
```

```
IF NOT (#CITY = 'A' THRU 'E')
```

Informationen über den Vergleich von Arrays in einem logischen Ausdruck finden Sie in **Verarbeitung von Arrays** (Seite 89).

Anmerkung:

Wenn mehrere logical-condition-criteria mit "AND" verknüpft werden, wird die Auswertung beendet, sobald das erste dieser Kriterien gefunden wird, das nicht erfüllt ist.

Arithmetische Operationen

- Initialisierung von Feldern
- Datenübertragung
- Abschneiden und Runden von Feldwerten
- Format/Länge von Ergebnisfeldern bei arithmetischen Operationen
- Arithmetische Operationen mit Gleitkomma-Zahlen
- Arithmetische Operationen mit Datum und Zeit
- Formatwahl im Hinblick auf die Verarbeitungszeit
- Genauigkeit von Ergebnissen bei arithmetischen Operationen
- Fehlerbedingungen bei arithmetischen Operationen
- Verarbeitung von Arrays.

Initialisierung von Feldern

Ein Feld — ob Datenbankfeld oder Benutzervariable —, das in einer arithmetischen Operation als Operand verwendet werden soll, muß mit einem der folgenden Formate definiert werden: N, P, I, F, D, T.

Anmerkung für Reporting Mode:

Ein Feld, das in einer arithmetischen Operation als Operand verwendet werden soll, muß vorher definiert werden.

Benutzervariablen oder Datenbankfelder, die in einer arithmetischen Operation als Ergebnisfeld verwendet werden, müssen nicht vorher definiert werden.

Sobald ein Programm zur Ausführung aufgerufen wird, werden alle im DEFINE DATA-Bereich definierten Benutzervariablen und Datenbankfelder mit den entsprechenden Leer- bzw. Nullwerten initialisiert.

Datenübertragung

Um Daten in ein anderes Feld zu übertragen, werden die Statements MOVE und COMPUTE verwendet. Die folgende Tabelle zeigt, zwischen welchen Formaten die Übertragung von Daten möglich ist, und sie bezieht sich auf die Übertragungsregeln:

Ausgangsfeld	Zielfeld											
	N / P	A	Bn (n < 5)	Bn (n > 4)	I	L	C	D	T	F	GH	OH
N oder P	✓	2	3	-	✓	-	-	-	✓	✓	-	-
A	-	✓	1	1	-	-	-	-	-	-	-	-
Bn (n < 5)	4	2	5	5	✓	-	-	-	✓	✓	-	-
Bn (n > 4)	-	6	5	5	-	-	-	-	-	-	-	-
I	✓	2	3	-	✓	-	-	-	✓	✓	-	-
L	-	9	-	-	-	✓	-	-	-	-	-	-
C	-	-	-	-	-	-	✓	-	-	-	-	-
D	✓	9	✓	-	✓	-	-	✓	7	✓	-	-
T	✓	9	✓	-	✓	-	-	8	✓	✓	-	-
F	✓	9 10	3	-	✓	-	-	-	✓	✓	-	-
GH	-	-	-	-	-	-	-	-	-	-	✓	-
OH	-	-	-	-	-	-	-	-	-	-	-	✓

✓ Übertragung möglich.

- Übertragung nicht möglich.

1... Siehe entsprechende Übertragungsregel auf Seite 77.

GH = GUI Handle, OH = Object Handle.

Siehe auch **Usage of Dynamic Variables** im *User's Guide for Windows NT*.

Konvertierung von Daten in ein anderes Format

Bei der Konvertierung von Werten in ein anderes Format gelten folgende Regeln:

- ① **Von Alphanumerisch (A) in Binär (B):** der Wert wird Byte für Byte von links nach rechts übertragen. Je nach Länge des Zielfeldes und der Anzahl der Bytes wird der Wert entweder abgeschnitten oder der Rest des Feldes mit Leerzeichen aufgefüllt.
- ② **Von Numerisch (N), Gepackt (P), Ganzzahl (I) und Binär mit 1–4 Bytes Länge (B) in Alphanumerisch (A):** Der Wert wird in ungepacktes Format umgesetzt und linksbündig in das Zielfeld übertragen, wobei vorangestellte Nullen weggelassen werden und der Rest des Feldes mit Leerzeichen aufgefüllt wird. Bei negativen numerischen Werten wird das Vorzeichen in die Hexadezimalnotation “Dx” umgesetzt. Ein Komma (Dezimalpunkt) im Ausgangswert wird nicht berücksichtigt, und alle Stellen vor und nach dem Komma werden als ganze Zahl interpretiert.
- ③ **Von Numerisch (N), Gepackt (P), Ganzzahl (I), Gleitkomma (F) in Binär mit 1–4 Bytes Länge (B):** Der Wert wird in binäres Format umgesetzt (4 Bytes). Ein Komma (Dezimalpunkt) wird ignoriert, die Stellen vor und nach dem Komma werden als ganze Zahl behandelt. Je nach Vorzeichen ist die Binärzahl entweder positiv oder das Zweierkomplement des Wertes.
- ④ **Von Binär mit 1–4 Bytes Länge (B) in Numerisch (N):** Der Wert wird umgesetzt und rechtsbündig übertragen, der Rest des Feldes wird mit Nullen aufgefüllt. Binäre Werte von 1–3 Bytes Länge werden immer als positiv interpretiert. Bei 4 Byte langen binären Werten bestimmt das erste (linke) Bit das Vorzeichen: 1 = negativ, 0 = positiv. Ein Komma (Dezimalpunkt) im Zielfeld wird nicht berücksichtigt, und alle Stellen vor und nach dem Komma werden als ganze Zahl interpretiert.
- ⑤ **Von Binär (B) in Binär (B):** Der Wert wird Byte für Byte von rechts nach links übertragen, und der Rest des Feldes wird mit Nullen aufgefüllt.
- ⑥ **Von Binär mit mehr als 4 Bytes (B) in Alphanumerisch (A):** Der Wert wird Byte für Byte von links nach rechts übertragen. Je nach Länge des Zielfeldes und der Anzahl der Bytes wird der Wert entweder abgeschnitten oder der Rest des Feldes mit Leerzeichen aufgefüllt.
- ⑦ **Von Datum (D) in Zeit (T):** Das Datum wird in Zeit umgesetzt, und zwar ausgehend von der Zeit 00:00:00:0.
- ⑧ **Von Zeit (T) in Datum (D):** Die Zeitkomponente wird abgeschnitten und nur die Datumskomponente des Zeitfeldes wird in das Datumsfeld übertragen.
- ⑨ **Von Logisch (L), Datum (D), Zeit (T), Gleitkomma (F) in Alphanumerisch (A):** Der Wert wird in Anzeigeform umgesetzt und linksbündig übertragen.
- ⑩ Wird ein Gleitkomma-Wert in ein alphanumerisches Feld übertragen, das zu kurz ist, wird die Mantisse entsprechend gekürzt.



Natural Referenzhandbuch

Siehe auch **Usage of Large and Dynamic Variables/Fields** im *User's Guide for Windows NT*.

Abschneiden und Runden von Feldwerten

Ist ein Feld zu kurz, wird der Feldwert abgeschnitten. Hierbei gilt folgendes:

- Numerische Felder: vorangestellte Stellen dürfen nur abgeschnitten werden, falls ihr Wert Null ist. Stellen nach einem ausdrücklich angegebenen oder implizierten Komma (Dezimalpunkt) dürfen abgeschnitten werden.
- Alphanumerische Felder: Nachfolgende Stellen dürfen abgeschnitten werden.

Aufrunden von Feldwerten:

- Bei Verwendung der Option **ROUNDED** wird die letzte Stelle im Feld aufgerundet, falls die erste abgeschnittene Stelle größer/gleich 5 ist.
Zur Ergebnisgenauigkeit einer Division siehe auch **Genauigkeit von Ergebnissen bei arithmetischen Operationen** (Seite 87).

Format/Länge von Ergebnisfeldern bei arithmetischen Operationen

Die folgende Tabelle zeigt Format/Länge von Ergebnisfeldern bei arithmetischen Operationen:

	I1	I2	I4	N oder P	F4	F8
I1	I1	I2	I4	P*	F4	F8
I2	I2	I2	I4	P*	F4	F8
I4	I4	I4	I4	P*	F4	F8
N oder P	P*	P*	P*	P*	F4	F8
F4	F4	F4	F4	F4	F4	F8
F8	F8	F8	F8	F8	F8	F8

P* ergibt sich aus der ganzzahligen Länge und Genauigkeit der einzelnen Operanden je nach Operation (siehe Abschnitt **Genauigkeit von Ergebnissen bei arithmetischen Operationen** (Seite 87)).

Für Format I gelten folgende dezimale Ganzzahl-Längen und möglichen Werte:

Format/Länge	Dezimale Ganzzahl-Länge	Mögliche Werte
I1	3	-128 bis 127
I2	5	-32 768 bis 32 767
I4	10	-2 147 483 648 bis 2 147 483 647

Arithmetische Operationen mit Gleitkomma-Zahlen

Einige allgemeine Hinweise

Gleitkomma-Zahlen (Format F) werden (ebenso wie Ganzzahlen (Format I)) als Summe von Zweierpotenzen dargestellt, wohingegen ungepackte und gepackte Zahlen (Formate N und P) als Summe von Zehnerpotenzen dargestellt werden.

Bei ungepackten und gepackten Zahlen ist die Position des Dezimalkommas fest. Bei Gleitkomma-Zahlen dagegen ist (wie der Name schon andeutet) die Position des Dezimalkommas "gleitend", d.h. seine Position ist nicht fest, sondern hängt vom tatsächlichen Wert der Zahl ab.

Gleitkomma-Zahlen sind unverzichtbar bei der Berechnung trigonometrischer und mathematischer Funktionen wie etwa Sinus oder Logarithmus.

Die Genauigkeit von Gleitkomma-Zahlen

Die Genauigkeit von Gleitkomma-Zahlen ist per se begrenzt:

- Bei einer Variablen von Format/Länge F4 ist die Genauigkeit auf etwa 7 Stellen begrenzt.
- Bei einer Variablen von Format/Länge F8 ist die Genauigkeit auf etwa 15 Stellen (bzw. 16 Stellen auf Großrechnern) begrenzt.

Werte mit einer größeren Anzahl signifikanter Stellen lassen sich nicht exakt als Gleitkomma-Zahlen darstellen. Unabhängig von der Zahl zusätzlicher Vor- oder Nachkommastellen kann eine Gleitkomma-Zahl nur die ersten 7 bzw. 15 (16) Stellen abdecken.

Eine Ganzzahl läßt sich nur exakt in einer Variablen von Format/Länge F4 darstellen, wenn ihr absoluter Wert nicht größer als $2^{23} - 1$ (bzw. $2^{24} - 1$ auf Großrechnern) ist.

Konvertierung in Gleitkomma-Darstellung

Wenn ein alphanumerischer, ungepackter numerischer oder gepackter numerischer Wert in Gleitkomma-Format umgesetzt wird (zum Beispiel bei einer Zuweisung), muß auch die Darstellungsform geändert werden, d.h. eine Summe von Zehnerpotenzen muß in eine Summe von Zweierpotenzen konvertiert werden.

Folglich lassen sich nur Zahlen, die als endliche Summe von Zweierpotenzen darstellbar sind, exakt darstellen; alle anderen Zahlen lassen sich nur näherungsweise darstellen.

Beispiele:

Diese Zahl hat eine exakte Gleitkomma-Darstellung:

$$1.25 = 2^0 + 2^{-2}$$

Diese Zahl ist eine periodische Gleitkomma-Zahl ohne exakte Darstellung:

$$1.2 = 2^0 + 2^{-3} + 2^{-4} + 2^{-7} + 2^{-8} + 2^{-11} + 2^{-12} + \dots$$

Daher kann die Konvertierung von alphanumerischen, ungepackt numerischen oder gepackt numerischen Werten in Gleitkomma-Werte, und umgekehrt, zu kleineren Fehlern führen.

Plattform-abhängige Unterschiede

Wie bereits durch einige der oben erwähnten unterschiedlichen Höchstwerte angedeutet, ist die Darstellung von Gleitkomma-Zahlen auf Großrechnern anders als auf anderen Plattformen.

Dadurch erklärt sich, warum dieselbe Anwendung mit Gleitkomma-Arithmetik leicht unterschiedliche Ergebnisse liefern kann, wenn sie auf verschiedenen Plattformen läuft.

Wenn Sie eine Natural-Anwendung auf eine andere Plattform portieren, beachten Sie bitte auch, daß der mögliche Wertebereich für Gleitkomma-Variablen auf einem Großrechner anders ist als auf anderen Plattformen:

- Der mögliche Wertebereich auf einem Großrechner ist für F4- und F8-Variablen (ca.):
 $\pm 5.4 * 10^{-79}$ bis $\pm 7.2 * 10^{75}$
- Der mögliche Wertebereich auf den meisten anderen Plattformen ist (ca.):
 für F4-Variablen: $\pm 1.17 * 10^{-38}$ bis $\pm 3.40 * 10^{38}$
 für F8-Variablen: $\pm 2.22 * 10^{-308}$ bis $\pm 1.79 * 10^{308}$

Anmerkung:

Die von Ihrem Taschenrechner verwendete Darstellung kann sich ebenfalls von der Ihres Computers unterscheiden — und die Ergebnisse für die gleiche Berechnung können daher auch hier unterschiedlich sein.

Arithmetische Operationen mit Datum und Zeit

Mit Feldern der Formate D (Datum) und T (Time = Zeit) sind nur Addition und Subtraktion erlaubt; Multiplikation und Division sind nicht erlaubt.

Datums-/Zeitwerte können zueinander addiert bzw. voneinander subtrahiert werden; oder Ganzzahl-Werte (ohne Nachkommastellen) können zu/von Datums-/Zeitwerten hinzuaddiert/subtrahiert werden. Solche ganzzahligen Werte können in Feldern der Formate N, P, I, D oder T enthalten sein.

Von ganzzahligen Werten, die zu einem Datumswert hinzuaddiert oder von einem Datumswert subtrahiert werden, wird angenommen, daß es sich um Tage handelt. Von ganzzahligen Werten, die zu einem Zeitwert hinzuaddiert oder von einem Zeitwert subtrahiert werden, wird angenommen, daß es sich um Zehntelsekunden handelt.

Bei arithmetischen Operationen mit Datum und Zeit gelten gewisse Einschränkungen, und zwar aufgrund von Naturals interner Behandlung von Datums- und Zeitarithmetik, wie im folgenden erläutert.

Intern behandelt Natural eine arithmetische Operation mit Datums- bzw. Zeitvariablen wie folgt:

COMPUTE *ergebnisfeld* = *operand1* +/- *operand2*

Das obige Statement wird aufgelöst als:

$$\textcircled{1} \quad \textit{zwischenenergebnis} = \textit{operand1} \textit{ +/- } \textit{operand2}$$

$$\textcircled{2} \quad \textit{ergebnisfeld} = \textit{zwischenenergebnis}$$

Das heißt, zunächst berechnet Natural das Ergebnis der Addition/Subtraktion, und erst danach weist es das Ergebnis dem Ergebnisfeld zu.

Komplexere arithmetische Operationen werden nach dem gleichen Muster aufgelöst:

COMPUTE *ergebnisfeld* = *operand1* +/- *operand2* +/- *operand3* +/- *operand4*

Das obige Statement wird aufgelöst als:

$$\textcircled{1} \quad \textit{zwischenenergebnis1} = \textit{operand1} \textit{ +/- } \textit{operand2}$$

$$\textcircled{2} \quad \textit{zwischenenergebnis2} = \textit{zwischenenergebnis1} \textit{ +/- } \textit{operand3}$$

$$\textcircled{3} \quad \textit{zwischenenergebnis3} = \textit{zwischenenergebnis2} \textit{ +/- } \textit{operand4}$$

$$\textcircled{4} \quad \textit{ergebnisfeld} = \textit{zwischenenergebnis3}$$

Das interne Format von einem solchen *zwischenenergebnis* hängt vom Format der einzelnen Operanden ab, wie die folgenden Tabellen zeigen.

Die folgende Tabelle zeigt das Format vom *zwischenenergebnis* einer **Addition** ($\text{zwischenenergebnis} = \text{operand1} + \text{operand2}$):

Format von <i>operand1</i>	Format von <i>operand2</i>	Format von <i>zwischenenergebnis</i>
D	D	Di
D	T	T
D	N, P, I	D
T	D, T, N, P, I	T
N, P, I	D	D
N, P, I	T	T

Die folgende Tabelle zeigt das Format vom *zwischenenergebnis* einer **Subtraktion** ($\text{zwischenenergebnis} = \text{operand1} - \text{operand2}$):

Format von <i>operand1</i>	Format von <i>operand2</i>	Format von <i>zwischenenergebnis</i>
D	D	Di
D	T	Ti
D	N, P, I	D
T	D, T	Ti
T	N, P, I	T
N, P, I	D	Di
N, P, I	T	Ti

Di ist ein Wert im internen Datumsformat; **Ti** ist ein Wert im internen Zeitformat; solche Werte können zwar in weiteren arithmetischen Datums-/Zeitoperationen verwendet werden, aber sie können keinem Ergebnisfeld vom Format **D** zugewiesen werden (siehe Zuweisungstabelle unten).

Bei komplexen arithmetischen Operationen, bei denen ein Zwischenergebnis im internen Format **Di** bzw. **Ti** als Operand für eine weitere Addition/Subtraktion verwendet wird, wird davon ausgegangen, daß es Format **D** bzw. **T** hat.

Die folgende Tabelle zeigt, welche Zwischenergebnisse intern welchen Ergebnisfeldern zugewiesen werden können (*ergebnisfeld* = *zwischenenergebnis*).

Format von <i>ergebnisfeld</i>	Format von <i>zwischenenergebnis</i>	Zuweisung möglich
D	D, T	ja
D	Di, Ti, N, P, I	<i>nein</i>
T	D, T, Di, Ti, N, P, I	ja
N, P, I	D, T, Di, Ti, N, P, I	ja

Ein Ergebnisfeld vom Format D oder T darf keinen negativen Wert enthalten.

Beispiele 1 und 2 (ungültig):

```
COMPUTE DATE1 (D) = DATE2 (D) + DATE3 (D)
```

```
COMPUTE DATE1 (D) = DATE2 (D) - DATE3 (D)
```

Diese Operationen sind nicht möglich, da das Zwischenergebnis der Addition bzw. Subtraktion Format *Di* hätte, und ein Wert vom Format *Di* keinem Ergebnisfeld vom Format D zugewiesen werden kann.

Beispiele 3 und 4 (ungültig):

```
COMPUTE DATE1 (D) = TIME2 (T) - TIME3 (T)
```

```
COMPUTE DATE1 (D) = DATE2 (D) - TIME3 (T)
```

Diese Operationen sind nicht möglich, da das Zwischenergebnis der Addition bzw. Subtraktion Format *Ti* hätte, und ein Wert vom Format *Ti* keinem Ergebnisfeld vom Format D zugewiesen werden kann.

Beispiel 5 (gültig):

```
COMPUTE DATE1 (D) = DATE2 (D) - DATE3 (D) + TIME3 (T)
```

Diese Operation ist möglich. Zunächst wird DATE3 von DATE2 subtrahiert, woraus sich ein Zwischenergebnis vom Format *Di* ergibt; dann wird dieses Zwischenergebnis zu TIME3 hinzuaddiert, woraus sich ein Zwischenergebnis vom Format T ergibt; und schließlich wird dieses zweite Zwischenergebnis dem Ergebnisfeld DATE1 zugewiesen.

Wenn Sie einen Format-T-Wert einem Format-D-Feld zuweisen, müssen Sie dafür sorgen, daß der Zeitwert eine gültige Datumskomponente enthält.

Formatwahl im Hinblick auf die Verarbeitungszeit

Bei arithmetischen Operationen hat die Wahl der richtigen Feldformate starken Einfluß auf die Verarbeitungszeit:

Bei kaufmännischen Berechnungen unter OpenVMS, UNIX und Windows empfiehlt es sich, nur Felder mit dem Format I (Integer/Ganzzahl) zu verwenden.

Wenn Ihr Computer mit einem "Math-Koprozessor" ausgestattet ist, ist Format F (Floating Point/Gleitkomma) schneller als Format N oder P und fast genauso schnell wie Format I. Ohne "Math-Koprozessor" ist Format F ungefähr so langsam wie Format N oder P.

Bei kaufmännischen Berechnungen auf Großrechnern empfiehlt es sich, nur Felder mit dem Format P (numerisch gepackt) zu verwenden. Die Anzahl der Stellen hinter dem Komma (Dezimalpunkt) sollte möglichst für alle Operanden einheitlich gewählt werden.

Bei wissenschaftlichen Berechnungen empfiehlt es sich, nur Felder mit dem Format F (Gleitkomma-Format) zu verwenden.

Werden die numerischen Formate N und P mit dem Format F vermischt, erfolgt intern eine Umsetzung in Format F; diese Umsetzung führt zu einer nicht unbeträchtlichen CPU-Beanspruchung. Daher sollte es möglichst vermieden werden, bei arithmetischen Operationen unterschiedliche Formate zu verwenden.

Genauigkeit von Ergebnissen bei arithmetischen Operationen

Operation	Stellen vor dem Komma	Stellen nach dem Komma
Addition/Subtraktion	$A_v + 1$ oder $B_v + 1$ (das jeweils größere)	A_n oder B_n (das jeweils größere)
Multiplikation	$A_v + B_v + 2$	$A_n + B_n$ (höchstens 7)
Division	$A_v + B_n$	(siehe unten)
Potenzierung	$15 - A_n$ Ausnahme (auf Großrechnern): wenn die Hochzahl eine oder mehrere Stellen hinter dem Komma (Dezimalpunkt) aufweist; (auf allen anderen Plattformen): die Potenzierung wird allgemein intern in Gleitkomma-Format ausgeführt. Weitere Informationen siehe Arithmetische Operationen mit Gleitkomma-Zahlen.	A_n
Quadratwurzel	A_v	A_n
<p>A = Erster Operand B = Zweiter Operand E = Ergebnis v = Stellen vor dem Komma (Dezimalpunkt) n = Stellen nach dem Komma (Dezimalpunkt)</p>		

Nachkommastellen bei Divisionsergebnissen

Die Genauigkeit des Ergebnisses einer Division hängt davon ab, ob ein Ergebnisfeld vorhanden ist oder nicht:

- Ist ein Ergebnisfeld vorhanden, ist die Genauigkeit: **En** oder **An** (das jeweils größere) *.
- Ist kein Ergebnisfeld vorhanden, ist die Genauigkeit: **An** oder **Bn** (das jeweils größere) *.

Ein Ergebnisfeld *ist* vorhanden (bzw. wird als vorhanden angenommen) in einem COMPUTE- und DIVIDE-Statement sowie in einer logischen Bedingung, in der die Division *hinter* dem Vergleichsoperator steht (z.B.: IF #A = #B / #C THEN ...).

Ein Ergebnisfeld ist *nicht* vorhanden (bzw. wird als nicht vorhanden angenommen) in einer logischen Bedingung, in der die Division *vor* dem Vergleichsoperator steht (z.B.: IF #B / #C = #A THEN ...).

Ausnahme: Wenn Dividend und Divisor Ganzzahlen sind und mindestens eine davon eine Variable ist, dann ist auch das Divisionsergebnis eine Ganzzahl (unabhängig von der Genauigkeit des Ergebnisfeldes sowie der Verwendung der ROUNDED-Option).

* *Bei Verwendung der ROUNDED-Option erhöht sich die Ergebnisgenauigkeit intern um eine Stelle, bevor das Ergebnis tatsächlich gerundet wird.*

Fehlerbedingungen bei arithmetischen Operationen

Bei einer Addition, Subtraktion, Multiplikation oder Division erhalten Sie einen Fehler, wenn das Ergebnis (insgesamt, d.h. vor und nach dem Komma) mehr als 31 Stellen hat. Bei einer Potenzierung erhalten Sie unter einer der folgenden Bedingungen einen Fehler:

- wenn die Basis gepacktes Format hat und entweder das Ergebnis mehr als 16 Stellen oder ein Zwischenergebnis mehr als 15 Stellen hat;
- wenn die Basis Gleitkomma-Format hat und das Ergebnis größer ist als ca. $7 * 10^{75}$.

Verarbeitung von Arrays

Grundsätzlich gelten folgende Regeln:

- Alle Skalar-Operationen können auf Array-Elemente, die aus einer einzigen Ausprägung bestehen, angewandt werden.
- Wenn eine Variable mit einem konstanten Wert definiert ist (zum Beispiel: #FELD (12) CONSTANT <8>), dann wird der Wert der Variablen bei der Kompilierung zugewiesen, und die Variable wird als Konstante behandelt. Falls eine solche Variable in einem Array-Index verwendet wird, bedeutet dies, daß die betreffende Dimension eine *bestimmte* Anzahl von Ausprägungen hat.
- Bei einer Zuweisung bzw. einem Vergleich zwischen zwei Arrays mit unterschiedlich vielen Dimensionen wird angenommen, daß die “fehlende” Dimension in dem Array mit weniger Dimensionen (1:1) ist.

Beispiel:

Wenn das Array #ARRAY1 (1:2) dem Array #ARRAY2 (1:2,1:2) zugewiesen wird, wird für #ARRAY1 angenommen, daß es #ARRAY1 (1:1,1:2) ist.

Zuweisungen mit Arrays

Wenn Sie einen Array-Bereich einem anderen Array-Bereich zuweisen, erfolgt die Zuweisung Element für Element.

Wenn Sie eine einzelne Ausprägung einem Array-Bereich zuweisen, wird jedes Element des Bereiches mit dem Wert der einzelnen Ausprägung gefüllt. (Bei einer mathematischen Funktion wird jedes Element des Bereiches mit dem Ergebnis der Funktion gefüllt.)

Bevor eine Zuweisung ausgeführt wird, werden die einzelnen Dimensionen der betroffenen Arrays miteinander verglichen, um zu prüfen, ob sie eine der unten aufgeführten Bedingungen erfüllen.

Die Dimensionen werden dabei unabhängig voneinander verglichen; d.h. die 1. Dimension des einen Arrays wird mit der 1. Dimension des anderen Arrays verglichen, die 2. Dimension des einen Arrays wird mit der 2. Dimension des anderen Arrays verglichen, und die 3. Dimension des einen Arrays wird mit der 3. Dimension des anderen Arrays verglichen.

Die Zuweisung von Werten eines Arrays an ein anderes Array ist nur unter einer der folgenden Bedingungen erlaubt:

- Die beiden verglichenen Dimensionen haben die gleiche Anzahl von Ausprägungen.
- Die beiden verglichenen Dimensionen haben beide eine unbestimmte Anzahl von Ausprägungen.
- Die Dimension, die einer anderen Dimension zugewiesen wird, besteht aus einer einzelnen Ausprägung.

Das folgende Programm zeigt, welche Zuweisungen zwischen Arrays möglich sind:

Beispiel — Array-Zuweisungen:

```

DEFINE DATA LOCAL
1 A1 (N1/1:8)
1 B1 (N1/1:8)
1 A2 (N1/1:8,1:8)
1 B2 (N1/1:8,1:8)
1 A3 (N1/1:8,1:8,1:8)
1 I (I2) INIT <4>
1 J (I2) INIT <8>
1 K (I2) CONST <8>
END-DEFINE
*
COMPUTE A1(1:3) = B1(6:8) /* allowed
COMPUTE A1(1:I) = B1(1:I) /* allowed
COMPUTE A1(*) = B1(1:8) /* allowed
COMPUTE A1(2:3) = B1(I:I+1) /* allowed
COMPUTE A1(1) = B1(I) /* allowed
COMPUTE A1(1:I) = B1(3) /* allowed
COMPUTE A1(I:J) = B1(I+2) /* allowed
COMPUTE A1(1:I) = B1(5:J) /* allowed
COMPUTE A1(1:I) = B1(2) /* allowed
COMPUTE A1(1:2) = B1(1:J) /* NOT ALLOWED (NAT0631)
COMPUTE A1(*) = B1(1:J) /* NOT ALLOWED (NAT0631)
COMPUTE A1(*) = B1(1:K) /* allowed
COMPUTE A1(1:J) = B1(1:K) /* NOT ALLOWED (NAT0631)
*
COMPUTE A1(*) = B2(1,*) /* allowed
COMPUTE A1(1:3) = B2(1,I:I+2) /* allowed
COMPUTE A1(1:3) = B2(1:3,1) /* NOT ALLOWED (NAT0631)
*
COMPUTE A2(1,1:3) = B1(6:8) /* allowed
COMPUTE A2(*,1:I) = B1(5:J) /* allowed
COMPUTE A2(*,1) = B1(*) /* NOT ALLOWED (NAT0631)
COMPUTE A2(1:I,1) = B1(1:J) /* NOT ALLOWED (NAT0631)
COMPUTE A2(1:I,1:J) = B1(1:J) /* allowed
*
COMPUTE A2(1,I) = B2(1,1) /* allowed
COMPUTE A2(1:I,1) = B2(1:I,2) /* allowed
COMPUTE A2(1:2,1:8) = B2(I:I+1,*) /* allowed
*
COMPUTE A3(1,1,1:I) = B1(1) /* allowed
COMPUTE A3(1,1,1:J) = B1(*) /* NOT ALLOWED (NAT0631)
COMPUTE A3(1,1,1:I) = B1(1:I) /* allowed
COMPUTE A3(1,1:2,1:I) = B2(1,1:I) /* allowed
COMPUTE A3(1,1,1:I) = B2(1:2,1:I) /* NOT ALLOWED (NAT0631)
END

```

Vergleiche mit Arrays

Grundsätzlich gilt folgendes: wenn mehrdimensionale Arrays miteinander verglichen werden, werden die einzelnen Dimensionen unabhängig voneinander behandelt; d.h. die 1. Dimension des einen Arrays wird mit der 1. Dimension des anderen Arrays verglichen, die 2. Dimension des einen Arrays wird mit der 2. Dimension des anderen Arrays verglichen, und die 3. Dimension des einen Arrays wird mit der 3. Dimension des anderen Arrays verglichen.

Der Vergleich zweier Array-Dimensionen ist nur unter einer der folgenden Bedingungen erlaubt:

- Die beiden verglichenen Dimensionen haben die gleiche Anzahl von Ausprägungen.
- Die beiden verglichenen Dimensionen haben beide eine unbestimmte Anzahl von Ausprägungen.
- Alle Dimensionen des einen Arrays bestehen jeweils aus einer einzelnen Ausprägung.

Das folgende Programm zeigt, welche Vergleiche zwischen Arrays möglich sind:

Beispiel — Array-Vergleiche:

```

DEFINE DATA LOCAL
1 A3 (N1/1:8,1:8,1:8)
1 A2 (N1/1:8,1:8)
1 A1 (N1/1:8)
1 I (I2) INIT <4>
1 J (I2) INIT <8>
1 K (I2) CONST <8>
END-DEFINE
*
IF A2(1,1) = A1(1) THEN IGNORE END-IF /* allowed
IF A2(1,1) = A1(I) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A1(1) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A1(I) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A1(*) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A1(I -3:I+4) THEN IGNORE END-IF /* allowed
IF A2(1,5:J) = A1(1:I) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A1(1:I) THEN IGNORE END-IF /* NOT ALLOWED(NAT0629)
IF A2(1,*) = A1(1:K) THEN IGNORE END-IF /* allowed
*
IF A2(1,1) = A2(1,1) THEN IGNORE END-IF /* allowed
IF A2(1,1) = A2(1,I) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A2(1,1:8) THEN IGNORE END-IF /* allowed
IF A2(1,*) = A2(I,I -3:I+4) THEN IGNORE END-IF /* allowed
IF A2(1,1:I) = A2(1,I+1:J) THEN IGNORE END-IF /* allowed
IF A2(1,1:I) = A2(1,I:I+1) THEN IGNORE END-IF /* NOT ALLOWED(NAT0629)
IF A2(*,1) = A2(1,*) THEN IGNORE END-IF /* NOT ALLOWED(NAT0629)
IF A2(1,1:I) = A1(2,1:K) THEN IGNORE END-IF /* NOT ALLOWED(NAT0629)
*
IF A3(1,1,*) = A2(1,*) THEN IGNORE END-IF /* allowed
IF A3(1,1,*) = A2(1,I -3:I+4) THEN IGNORE END-IF /* allowed
IF A3(1,*,I:J) = A2(*,1:I+1) THEN IGNORE END-IF /* allowed
IF A3(1,*,I:J) = A2(*,I:J) THEN IGNORE END-IF /* allowed
END

```

Wenn Sie zwei Array-Bereiche miteinander vergleichen, beachten Sie bitte, daß die folgenden zwei Ausdrücke zu unterschiedlichen Ergebnissen führen:

```
#ARRAY1(*) NOT EQUAL #ARRAY2(*)
```

```
NOT #ARRAY1(*) = #ARRAY2(*)
```

Beispiel:

Bedingung A:

```
IF #ARRAY1(1:2) NOT EQUAL #ARRAY2(1:2)
```

Dies entspricht:

```
IF (#ARRAY1(1) NOT EQUAL #ARRAY2(1)) AND (#ARRAY1(2) NOT EQUAL #ARRAY2(2))
```

Bedingung A ist also erfüllt, wenn die erste Ausprägung von #ARRAY1 ungleich der ersten Ausprägung von #ARRAY2 ist *und* die zweite Ausprägung von #ARRAY1 ungleich der zweiten Ausprägung von #ARRAY2 ist.

Bedingung B:

```
IF NOT #ARRAY1(1:2) = #ARRAY2(1:2)
```

Dies entspricht:

```
IF NOT (#ARRAY1(1) = #ARRAY2(1) AND #ARRAY1(2) = #ARRAY2(2))
```

Dies wiederum entspricht:

```
IF (#ARRAY1(1) NOTEQUAL #ARRAY2(1)) OR (#ARRAY1(2) NOT EQUAL #ARRAY2(2))
```

Bedingung B ist also erfüllt, wenn *entweder* die erste Ausprägung von #ARRAY1 ungleich der ersten Ausprägung von #ARRAY2 ist *oder* die zweite Ausprägung von #ARRAY1 ungleich der zweiten Ausprägung von #ARRAY2 ist.

Arithmetische Operationen mit Arrays

In arithmetischen Operationen (in COMPUTE-, ADD- oder MULTIPLY-Statements) können Array-Bereiche auf folgende Arten verwendet werden:

- $\text{bereich} + \text{bereich} = \text{bereich}$.
Die Dimensionen der Bereiche müssen gleich sein.
Die Addition wird Element für Element ausgeführt.
- $\text{bereich} * \text{bereich} = \text{bereich}$.
Die Dimensionen der Bereiche müssen gleich sein.
Die Multiplikation wird Element für Element ausgeführt.
- $\text{skalar} + \text{bereich} = \text{bereich}$.
Die Dimensionen der Bereiche müssen gleich sein.
Der Skalarwert wird jedem Element des Bereichs hinzuaddiert.
- $\text{bereich} * \text{skalar} = \text{bereich}$.
Die Dimensionen der Bereiche müssen gleich sein.
Jedes Element des Bereichs wird mit dem Skalarwert multipliziert.
- $\text{bereich} + \text{skalar} = \text{skalar}$.
Jedes Element des Bereichs wird dem Skalarwert hinzuaddiert und das Ergebnis im Skalar ausgegeben.
- $\text{skalar} * \text{bereich} = \text{skalar2}$.
Der Skalarwert wird mit jedem Element des Bereichs multipliziert und das Ergebnis in Skalar2 ausgegeben.

Referenzen auf Sourcecode-Zeilennummern unnumerieren

Vierstellige, numerische und ein Statement referenzierende Sourcecode-Zeilennummern (siehe die Statement-Referenz-Notation – r) werden auch unnumeriert, wenn das Natural Source-Program unnumeriert wird. Zwecks größerer Benutzerfreundlichkeit, zur Erhöhung der Lesbarkeit und Unterstützung bei der Fehlerbeseitigung werden alle in einem Statement auftretenden Referenzen auf Sourcecode-Zeilennummern, eine alphanumerische Konstante oder ein Kommentar unnumeriert. Die Position der Referenz auf die Sourcecode-Zeilenummer im Statement oder der alphanumerischen Konstanten (Anfang, Mitte, Ende) spielt keine Rolle.

Die folgenden Muster werden als eine gültige Referenz auf eine Sourcecode-Zeilenummer erkannt und unnumeriert (*nnnn* ist eine vierstellige Zahl):

Muster	Beispiel-Statement
<i>(nnnn)</i>	ESCAPE BOTTOM (0150)
<i>(nnnn /</i>	DISPLAY ADDRESS-LINE (0010/1:5)
<i>(nnnn,</i>	DISPLAY NAME (0010,A10/1:5)

Wenn auf die linke Klammer oder die vierstellige Zahl *nnnn* ein Leerzeichen folgt, oder wenn auf die vierstellige Zahl *nnnn* ein Punkt folgt, wird das Muster nicht als gültige Referenz auf eine Sourcecode-Zeilenummer angesehen.

Um zu verhindern, daß eine vierstellige, in einer alphanumerischen Konstante enthaltene Zahl unabsichtlich unnumeriert wird, sollte die Konstante aufgespalten werden, und die unterschiedlichen Bestandteile sollten mittels eines Bindestrichs zu einem einzelnen Wert verkettet werden.

Beispiel:

```
Z := 'XXXX (1234,00) YYYY'
```

sollte ersetzt werden durch

```
Z := 'XXXX (1234' - ',00) YYYY'
```

SESSION-PARAMETER

Natural-Session-Parameter dienen dazu, bestimmte Faktoren zu beeinflussen, wie etwa Aspekte der Report-Formatierung, der Ausgabe von Feldern, usw.

Dieses Kapitel behandelt die folgenden Themen:

- Allgemeine Informationen
- Setzen von Session-Parameterwerten
- Verarbeitung von Session-Parametern
- Die Session-Parameter im Überblick.

Allgemeine Informationen

In Natural werden eine Reihe von Session-Parametern verwendet:

- um bestimmte Zeichen festzulegen
- um Verarbeitungszeitgrenzen festzulegen,
- um Reaktionen auf bestimmte Bedingungen festzulegen
- um bestimmte Maximal- bzw. Minimalgrößen festzulegen
- um bestimmte Aspekte der Reportformatierung festzulegen.

Im Abschnitt **Die Parameter im Überblick** (Seite 102) sowie in den Erklärungen zu den einzelnen Parametern ist angegeben, bei welchen Statements ein Session-Parameter jeweils gesetzt bzw. verarbeitet werden kann.

Bei der Installation von Natural legt der Natural-Administrator für diese Parameter bestimmte Standardwerte fest, die dann für alle Natural-Benutzer gelten.

Die für Ihre Natural-Session gültigen Parameterwerte können Sie sich ansehen, wenn Sie das Systemkommando `GLOBALS` eingeben. Nähere Einzelheiten hierzu finden Sie in Ihrem *Natural User's Guide* bzw. *Natural Benutzerhandbuch*.

Setzen von Parameterwerten

Es gibt verschiedene Möglichkeiten, die Werte der Session-Parameter zu setzen:

- über das Standard-Parametermodul bzw. die Standard-Parameterdatei (NATPARM), das/die bei der Installation von Natural angelegt wird;
- über dynamische Parameter beim Aufrufen einer Natural-Session (wie in Ihrer *Natural Reference Documentation* beschrieben);
- mit dem Systemkommando GLOBALS;
- mit dem Statement SET GLOBALS (nur im *Reporting Mode*);
- mit dem Statement FORMAT;
- mit Statements wie INPUT, DISPLAY oder WRITE, in denen bestimmte Parameter verarbeitet werden;
- über Terminalkommandos.

Statt der Parameterwerte “ON” und “OFF” können Sie wahlweise auch die Werte “T” (true = wahr) bzw. “F” (falsch) verwenden.

Ändern von Parameterwerten auf Session-Ebene — Das GLOBALS-Kommando

Einige der vom Natural-Administrator festgelegten Standardwerte können Sie für die Dauer einer Natural-Session ändern.

Sie können die Parameterwerte innerhalb einer Session ändern, indem Sie das folgende Systemkommando verwenden:

GLOBALS

Wenn Sie das GLOBALS-Kommando ausführen, erhalten Sie einen Schirm, auf dem die für Ihre laufende Session derzeit gültigen Parameterwerte angezeigt werden. Auf diesem Schirm können Sie auch die Werte ändern, die nicht Ihren Anforderungen entsprechen.

Die mit einem GLOBALS-Kommando festgelegten Parameterwerte gelten bis zum Ende der jeweiligen Natural-Session (und gelten für jedes im Laufe der Session gespeicherte Objekt), es sei denn, Sie ändern sie wieder durch ein anderes GLOBALS-Kommando.

Ändern von Parameterwerten auf Programm-Ebene — Das FORMAT-Statement

Einige Parameter-Standardwerte können Sie für die Dauer eines Programms (Reports) ändern. Dies geschieht mit einem FORMAT-Statement im jeweiligen Programm, welches die betreffenden session-weit gültigen Parameter überschreibt.

Beispiel für ein FORMAT-Statement:

```
FORMAT AL=10 HC=R
```

Die mit einem FORMAT-Statement angegebenen Parameterwerte gelten bis zum Ende des ausgeführten Programms, wenn sie nicht vorher im Programm durch ein anderes FORMAT-Statement geändert werden.

Sie können allerdings nur einen Teil der Session-Parameter programmspezifisch umsetzen. Andererseits gibt es eine Reihe von Parametern, die Sie nur auf Programm-Ebene setzen können, nicht aber auf Session-Ebene. Dies sind überwiegend Parameter, die sich auf die Formatierung eines Reports auswirken.

Ändern von Parameterwerten auf Statement-Ebene

Die meisten Parameter, die Sie mit einem FORMAT-Statement festlegen können, können Sie auch auf Statement-Ebene setzen, zum Beispiel bei einem DISPLAY-, WRITE-, INPUT- oder REINPUT-Statement, und zwar durch Einfügen des Parameters (in Klammern) hinter dem Statement-Namen.

Beispiel:

```
DISPLAY (SF=4) NAME JOB-TITLE CURR-CODE SALARY
```

Ein auf Statement-Ebene angegebener Parameterwert gilt nur für das jeweilige Statement, und zwar vor allen auf anderen Ebenen für diesen Parameter angegebenen Werten.

Ändern von Parameterwerten auf Feld-Ebene

Innerhalb eines WRITE-, DISPLAY-, INPUT- oder REINPUT-Statements können Sie bestimmte Parameterwerte noch feldspezifisch festlegen, indem Sie den Parameterwert unmittelbar hinter dem betreffenden Feld (in Klammern) angeben.

Beispiel:

```
DISPLAY NAME (AL=10) JOB-TITLE CURR-CODE SALARY
```

Dieser Wert gilt dann nur für dieses Feld, und zwar vor allen auf anderen Ebenen für diesen Parameter angegebenen Werten. Die feldspezifische Angabe von Parameterwerten ist allerdings nur für einen Teil der auf Statement-Ebene beeinflussbaren Parameter möglich.

Verarbeitung von Parametern

Session-Parameter, die in den Statements DISPLAY, FORMAT, PRINT, INPUT, REINPUT, WRITE, WRITE TITLE und WRITE TRAILER gesetzt werden, werden während der *Programmkompileierung* verarbeitet und sind daher in dem betreffenden Objektmodul enthalten.

Bei der Auswertung der Parameterwerte gilt folgende Hierarchie:

- ① für einzelne Felder/Elemente gesetzte Parameter (höchste Priorität)
- ② auf Statement-Ebene gesetzte Parameter
- ③ mit einem FORMAT-Statement gesetzte Parameter
- ④ die Standardwerte der Parameter (niedrigste Priorität)

Mit einem SET GLOBALS-Statement gesetzte Parameter bewirken eine Veränderung der *Laufzeit*-Umgebung. Sie gelten solange, bis sie mit einem weiteren SET GLOBALS-Statement (oder GLOBALS-Systemkommando) geändert werden.

Die Parameter im Überblick

- X** Parameter kann dynamisch / mit GLOBALS-Kommando / mit FORMAT-Statement angegeben werden.
S Parameter kann auf Statement-Ebene angegeben werden.
E Parameter kann auf Element-Ebene (Feld-Ebene) angegeben werden.
K Parameter wird zur Kompilierungszeit ausgewertet.
L Parameter wird zur Laufzeit ausgewertet.

	Standardwert	Dynam. Parameter	GLOBALS Kommando	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
AD					X	SE	SE	SE	SE	SE
AL	keiner				X	SE	SE		SE	SE
BX	keiner				X	SE	SE		SE	
CC	OFF	X	X	L						
CD	NE				X	SE	SE	SE	SE	SE
CF	%	X	X	L						
CO	OFF	X	X							
CV	keiner					SE	SE		SE	SE
DC	.	X	X	L						
DF	S				X	SE	SE		SE	SE
DFOUT	S	X	X	L						
DFSTACK	S	X	X	L						
DFTITLE	S	X	X	L						
DU	OFF	X	X	L						
DY	keiner					SE	SE		SE	SE
EJ	ON	X	X	L						
EM	keiner				X	SE	SE		SE	SE
ENDIAN	DEFAULT	X	X							
ES	OFF				X	S			S	
FC	Leerz.				X	SE				
FCDP	ON	X	X	L						
FL	keiner				X	SE	SE		SE	SE
FS	OFF	X	X	L						
GC	Leerz.				X	SE				
HC	C				X	SE				
HE	keiner						SE			
HW	ON				X	S				
IA	=	X	X	L						
IC	keiner				X	SE				
ID	,	X	X	L						
IM	F	X	X	L						
IP	ON				X		SE			
IS	OFF				X	SE			SE	

	Standardwert	Dynam. Parameter	GLOBALS Kommando	Statements						
				SET GLOBALS	FORMAT	DISPLAY	INPUT	REINPUT	WRITE	PRINT
KD	OFF				X					
LC	keiner				X	SE				
LE	OFF	X	X	K						
LS	physische	X	X	K	X	S	S		S	
LT	99999999	X	X	L						
MC	1				X	S	S		S	S
ML	T Mainframe B Windows, UNIX, OpenVMS	X	X							
MP	32767				X	S			S	S
MS	OFF				X		S			
MT	60 Sek.	X	X	L						
NC	OFF	X	X	L						
NL	keiner				X	SE	SE		SE	SE
OPF	ON	X	X	L						
PC	1				X	S	S		S	S
PD	50	X	X	L						
PM	keiner	X	X	K	X	SE	SE		SE	SE
PS	physische	X	X	KL	X	S	S		S	
REINP	ON	X	X	L						
SA	OFF	X	X	L						
SF	1	X	X	K	X	SE				
SG	ON				X	SE	SE		SE	SE
SL	72	X	X							
SM	OFF	X	X							
SYMGEN	OFF	X	X							
TC	keiner				X	SE				
TS	OFF	X	X							
UC	-				X	SE				
WH	OFF	X	X	L						
ZD	ON	X	X	L						
ZP	ON	X	X	K	X	SE	SE	SE	SE	SE

AD — Attribut-Definition

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
AD	siehe unten	siehe unten	CALLNAT DISPLAY FORMAT INPUT PERFORM PRINT REINPUT WRITE

Mit diesem Session-Parameter definieren Sie Feldattribute auf Feld- oder Statement-Ebene. Sie können gleichzeitig mehrere Attribute, in beliebiger Reihenfolge, angeben:

$AD = \left[\begin{array}{c} \underline{B} \\ \underline{C} \\ \underline{D} \\ \underline{I} \\ \underline{N} \\ \underline{U} \\ \underline{V} \\ \underline{Y} \end{array} \right] \left[\begin{array}{c} \underline{L} \\ \underline{R} \\ \underline{Z} \end{array} \right] \left[\begin{array}{c} \underline{A} \\ \underline{M} \\ \underline{O} \\ \underline{P} \end{array} \right] \left[\begin{array}{c} \underline{E} \\ \underline{F} \end{array} \right] \left[\begin{array}{c} \underline{G} \\ \underline{H} \end{array} \right] \left[\begin{array}{c} \underline{T} \\ \underline{W} \end{array} \right] ['c']$

Die einzelnen Werte sind auf den folgenden Seiten erklärt.

Standardwerte sind unterstrichen.

Bei der Ausrichtung der Feldwerte (zweite Wertespalte in obiger Syntax) ist der Standardwert

L für alphanumerische Felder und

R für numerische Felder.

Beispiele:

```
DISPLAY #FIELD A (AD=R)
```

```
INPUT #FIELD B (AD=M)
```

```
INPUT (AD=IM) #FIELD A #FIELD B
```

Feldanzeige

Wert	Bedeutung	Statements	Erklärung
B	blinkend (*)	alle	Der Feldwert wird blinkend angezeigt.
C	kursiv (*)	alle	Der Feldwert wird kursiv angezeigt.
D	default; normal	alle	Der Feldwert wird normal angezeigt, d.h. weder intensiviert noch sonstwie hervorgehoben.
I	intensiviert	alle	Der Feldwert wird intensiviert, d.h. hell erleuchtet angezeigt.
N	nicht sichtbar	alle	Ein eingegebener Feldwert wird nicht angezeigt.
U	unterstrichen	alle	Der Feldwert wird unterstrichen angezeigt.
V	invers (*)	alle	Der Feldwert wird invers angezeigt, d.h. in farblicher Umkehrung von Hintergrund und Feldwert.
Y	dynamische Attribute	INPUT DISPLAY WRITE	Feldattribute werden dynamisch über eine Kontrollvariable (Format C) zugewiesen.

* Die mit einem Stern (*) markierten Feldanzeige-Attribute sind an entsprechende Hardware-Voraussetzungen gebunden und werden zur Laufzeit ignoriert, falls diese Voraussetzungen nicht gegeben sind.

Ausrichtung der Feldwerte

Wert	Bedeutung	Statements	Erklärung
L	linksbündig	alle	Feldwerte werden linksbündig im Feld ausgegeben.
R	rechtsbündig	alle	Feldwerte werden rechtsbündig im Feld ausgegeben.
Z	vorangestellte Nullen (Z = zeros)	alle	Feld werte werden rechtsbündig im Feld ausgegeben, der Rest des Feldes wird mit Nullen aufgefüllt.

Feldklassen

Wert	Bedeutung	Statements	Erklärung
A	Eingabefeld, ungeschützt (A = accept input)	INPUT	Das Feld ist ein reines Eingabefeld. Ein Feldwert wird in Antwort auf ein INPUT-Statement eingegeben.
M	Aus- und Eingabefeld (M = modifiable)	INPUT CALLNAT	Das Feld ist ein modifizierbares Ausgabefeld. Ein Feldwert wird bei der Ausführung eines INPUT-Statements angezeigt; der ausgegebene Wert kann vom Benutzer überschrieben werden.
O	Ausgabefeld, geschützt (O = output)	INPUT CALLNAT	Das Feld ist ein reines Ausgabefeld. Ein Feldwert wird bei der Ausführung eines INPUT-Statements angezeigt; der ausgegebene Wert kann nicht überschrieben werden.
P	vorübergehend geschützt (P = protected)	INPUT REINPUT	Wird in Verbindung mit einer Kontrollvariablen (Format C), dem DY-Parameter und dem REINPUT-Statement verwendet.

Eingabezwang

Wert	Bedeutung	Statements	Erklärung
E	Eingabe erforderlich	INPUT	Als Antwort auf ein INPUT-Statement muß ein Feldwert eingegeben werden; andernfalls erscheint eine Fehlermeldung. Dies ist nur bei reinen Eingabefeldern (AD=A) relevant.
F	Eingabe nicht erforderlich	INPUT	Als Antwort auf ein INPUT-Statement kann ein Feldwert eingegeben werden, muß aber nicht.

Mindestlänge der Eingabewerte

Wert	Bedeutung	Statements	Erklärung
G	Wertlänge	INPUT	Ein als Antwort auf ein INPUT-Statement eingegebener Wert muß genauso lang sein wie das Feld, d.h. das Feld muß vollständig gefüllt werden. Dies ist nur bei reinen Eingabefeldern (AD=A) relevant.
H	Wertlänge	INPUT	Ein als Antwort auf ein INPUT-Statement eingegebener Wert darf kürzer sein als das Feld, d.h. das Feld muß nicht vollständig gefüllt werden.

Groß-/Kleinschreibung

Wert	Bedeutung	Statements	Erklärung
T	Kleinbuchstaben werden umgesetzt	INPUT	Ein eingegebener Wert wird automatisch in Großbuchstaben umgesetzt, d.h. es ist egal, ob ein Wert in Klein- oder Großbuchstaben eingegeben wird.
W	Kleinbuchstaben werden akzeptiert	INPUT	Es erfolgt keine Umsetzung von Klein- in Großbuchstaben, d.h. ein Wert wird so interpretiert, wie er eingegeben wird. Um AD=W einzuschalten, müssen Sie den Wert ON für den Natural-Profilparameter LC angeben.

Füllzeichen

Wert	Bedeutung	Statements	Erklärung
'c'	Füllzeichen	INPUT	Der unbeschriebene Teil eines Feldes (für reine Eingabefelder) wird bei der Anzeige mit dem Zeichen c gefüllt, wenn AD=A oder AD=M angegeben wird.

Anmerkung für Großrechner:

Wenn das Füllzeichen auf Leerzeichen (x'40') gesetzt wird, werden auffüllende Leerzeichen durch x'00' ersetzt, damit eine Einfügung von Zeichen ermöglicht wird, ohne daß zuvor der Rest im Eingabefeld gelöscht werden muß. In BS2000-Umgebungen werden x'00'-Zeichen als Punkte auf Terminals des Typs 97xx angezeigt. Ihre Erscheinungsform kann mittels der SIDA-Utility oder mit der Konfigurations-Utility der entsprechenden Terminal-Emulation geändert werden.

AL — Alphanumerische Länge der Ausgabe

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
AL	1 bis n (n = Wert des LS-Parameters minus 1)	keiner	DISPLAY FORMAT INPUT PRINT WRITE

Mit diesem Session-Parameter bestimmen Sie die Länge der Ausgabe alphanumerischer Felder: wenn diese Länge kürzer als die eigentliche Feldlänge ist, werden ausgegebene Feldwerte unter Umständen abgeschnitten.

Für Eingabefelder (AD=A oder AD=M) in INPUT-Statements sollte dieser Parameter nicht verwendet werden.

Eine für ein Feld definierte Editiermaske setzt den AL-Parameter für dieses Feld außer Kraft.

Beispiel:

```
FORMAT AL=20
```

BX — Box-Definition (“Einrahmung”)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
BX	T, B, L, R, ON, OFF	keiner	DISPLAY FORMAT INPUT WRITE

Dieser Session-Parameter gilt nur für Natural auf Großrechnern.

“Outlining” ist nur auf bestimmten Terminaltypen möglich (in der Regel auf solchen, die auch Doppelbyte-Zeichensätze unterstützen). Falls das verwendete Terminal kein Outlining unterstützt, wird dieser Parameter zur Laufzeit ignoriert.

Wenn Sie die Session-Parameter BX=L oder BX=R verwenden, sollten Sie die Natural-Bildschirm-Optimierung mittels des Profilparameters DSC=OFF oder mittels des Terminalkommandos %RO ausschalten.

“Outlining” ist die Möglichkeit, bestimmte Felder auf dem Bildschirm “eingerahmt” anzuzeigen. Diese Form der Anzeige ist eine weitere Möglichkeit, dem Benutzer Länge und Position von Feldern auf dem Schirm deutlich zu machen.

Mit diesem Parameter bestimmen Sie, welcher Teil des “Rahmens” angezeigt werden soll:

T	(Top) Oberer waagerechter Rand.
B	(Bottom) Unterer waagerechter Rand.
L	Linker senkrechter Rand.
R	Rechter senkrechter Rand.

Sie können die Werte in beliebiger Reihenfolge angeben.

Beispiel:

```
DISPLAY #FIELD1 (BX=RLT) /
        #FIELD2 (BX=TLRB)
```

BX=ON entspricht **BX=TBRL**.

BX=OFF bewirkt, daß die betreffenden Felder nicht “eingerahmt” werden.

Vgl. Terminalkommando %D=.

CC — Conditional Execution (Bedingte Programmausführung)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
CC	ON / OFF	OFF	SET GLOBALS

Dieser Parameter gilt nur für Natural auf Großrechnern.

Mit diesem Parameter bestimmen Sie, was geschehen soll, wenn bei der Kompilierung oder Ausführung eines Natural-Programms im Batch-Betrieb ein Fehler auftritt.

CC=ON	Die Eingabedaten für SYNIN und OBJIN werden gelöscht, und zwar bis zu einer Zeile, die mit “%%” an den beiden ersten Stellen beginnt, oder einer “End-of-File”-Bedingung. Folgen weitere Eingabedaten, so liest Natural nach der “%%”-Zeile weiter.
CC=OFF	Natural führt das nächste Programm (oder Kommando) im Eingabedatenstrom aus.

Wenn die Natural-Session beendet wird, so wird — falls ein Fehler auftritt — Return Code 4 über Register 15 an das aufrufende Programm übergeben (und zwar unabhängig von der Einstellung des CC-Parameters).

Beispiel:

```
SET GLOBALS CC=ON
```

CD — Color Definition (Farbdefinition)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
CD	BL = blau GR = grün NE = neutral PI = pink RE = rot (red) TU = türkis YE = gelb (yellow)	NE	DISPLAY FORMAT INPUT REINPUT PRINT WRITE

Mit diesem Parameter bestimmen Sie die Farbe, in der Felder angezeigt werden.

Anmerkung:

Falls kein Farb-Bildschirm verwendet wird, wird dieser Parameter zur Laufzeit ignoriert.

Beispiel:

```
INPUT (CD=RE) #A #B
```

CF — Steuerzeichen für Terminalkommandos

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
CF	jedes Sonderzeichen, OFF	%	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie das Steuerzeichen für Natural-Terminalkommandos, d.h. das Zeichen, mit dem die Terminalkommandos beginnen und das die Terminalkommandos als solche identifiziert.

Gilt bspw. CF=% (Standardzeichen), so muß das erste Zeichen eines Terminalkommandos das Prozentzeichen “%” sein.

Wenn Sie CF=OFF setzen, so steht kein Steuerzeichen für Terminalkommandos zur Verfügung. Terminalkommandos, die mit einem SET CONTROL-Statement angegeben werden, können jedoch nach wie vor ausgeführt werden. CF=OFF darf nur dynamisch beim Aufruf von Natural oder mit einem GLOBALS-Kommando angegeben werden, nicht mit einem SET GLOBALS-Statement.

Das mit dem CF-Parameter definierte Zeichen muß ein anderes sein als das mit dem HI-Parameter (Hilfe-Zeichen) oder IA-Parameter (INPUT-Zuweisungszeichen) definierte.

Anmerkung:

Das mit dem CF-Parameter definierte Zeichen sollte ein anderes sein als das mit dem DC-Parameter (Dezimalkomma) oder ID-Parameter (INPUT-Delimiterzeichen) definierte.

Beispiel:

```
SET GLOBALS CF=+
```

Anmerkung:

Im Map-Editor ist das Steuerzeichen für Terminalkommandos immer “%” (um Konflikte mit in Maps verwendeten Delimiterzeichen zu vermeiden), egal welches Zeichen Sie mit dem CF-Parameter definiert haben.

CO — Compiler Output (Kompilier-Ausgabe)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
CO	ON / OFF	OFF	SET GLOBALS

Dieser Parameter sollte nicht ohne vorherige Rücksprache mit dem Software AG-Support verwendet werden. Dieser Parameter steht nicht auf Großrechnern sondern nur auf UNIX/OpenVMS-Plattformen zur Verfügung.

Mit diesem Parameter können Sie bestimmen, ob bei der Kompilierung eines Natural-Objekts ein Compiler-Listing angezeigt werden soll.

Das Setzen von CO=ON führt zu einer beträchtlich längeren Kompilierungszeit, als wenn dieser Parameter nicht benutzt wird (CO=OFF). Insbesondere mit einem CATAL- Kommando dauert es ziemlich lange, bis alle Objekte kompiliert sind. Deshalb sollte dieser Parameter standardmäßig auf OFF gesetzt werden.

CO=ON	Ein Compiler-Listing wird auf dem Bildschirm angezeigt.
CO=OFF	Es wird kein Compiler-Listing angezeigt.

CV — Control Variable (Kontrollvariable)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
CV	siehe unten	keiner	DISPLAY INPUT PRINT WRITE

Mit diesem Session-Parameter wird eine Kontrollvariable referenziert.

Eine Kontrollvariable wird mit Format C definiert und dient dazu, Feldern dynamisch Attribute zuzuweisen.

Die folgenden Feldattribute können dynamisch verändert werden:

- Feldanzeige (B, C, D, I, N, U, V; siehe Session-Parameter AD, Seite 104).
- Feldschutz (P; siehe Session-Parameter AD, Seite 104).
- Farbe (BL, GR, NE, PI, RE, TU, YE; zur Erklärung der Farbcodes siehe Session-Parameter CD, Seite 111).

Beispiel:

```

DEFINE DATA LOCAL
1 #ATTR(C)
1 #A (N5)
END-DEFINE
...
MOVE (AD=I CD=RE) TO #ATTR
INPUT #A (CV=#ATTR)
...

```

Mittels einer Kontrollvariablen kann auch überprüft werden, ob während der Ausführung eines INPUT-Statements Feldinhalte verändert worden sind (vgl. **Modifizierte Kontrollvariablen**).

```
IF #ATTR MODIFIED ...
```

Anmerkung:

Wenn Sie den CV-Parameter auf Statement-Ebene und auf Feldebene angeben und die Kontrollvariable für das betreffende Feld leer ist, wird die Kontrollvariable für das Statement auch für das Feld genommen.

DC — Decimal Character (Dezimalstellenzeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
DC	jedes Zeichen (außer Ziffern)	. (Punkt)	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, welches Zeichen als Dezimalkomma bzw. Dezimalpunkt verwendet wird.

Das mit DC definierte Zeichen gilt für alle Angaben, in denen ein Dezimalkomma vorkommt, d.h. in Variablen, Konstanten und Editiermasken.

Das mit dem DC-Parameter definierte Zeichen muß ein anderes sein als das mit dem IA-Parameter (INPUT-Zuweisungszeichen) oder ID-Parameter (INPUT-Delimiterzeichen) definierte.

Anmerkung:

Das mit dem DC-Parameter definierte Zeichen sollte ein anderes sein als das mit dem CF-Parameter (Steuerzeichen für Terminalkommandos) oder HI-Parameter (Hilfe-Zeichen) definierte.

Beispiel:

```
SET GLOBALS DC=,
```

DF — Datumsformat

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
DF	S, I, L	S	COMPRESS DISPLAY FETCH FORMAT INPUT MOVE PRINT RUN STACK WRITE

Wenn der Wert eines Datumsfeldes in alphanumerisches Format umgesetzt wird (z.B. in einem MOVE-, DISPLAY-, WRITE- oder INPUT-Statement) und für die Umsetzung keine Editiermaske angegeben ist, wird das durch den Profilparameter DTFORM bestimmte Standarddatumsformat als Editiermaske genommen.

Dasselbe gilt bei der Eingabeauswertung einer in einem INPUT-Statement verwendeten Datumsvariablen: Wenn keine Editiermaske angegeben ist, wird die Eingabe entsprechend des durch den Profilparameter DTFORM bestimmten Datumsformats ausgewertet.

Mit dem DF-Parameter bestimmen Sie die Länge eines in alphanumerische Darstellung umgesetzten Datums, wenn hierfür keine Editiermaske angegeben ist:

DF=S	8-Byte-Darstellung mit 2-stelliger Jahreskomponente und Delimitern (<i>yy-mm-dd</i>).
DF=I	8-Byte-Darstellung mit 4-stelliger Jahreskomponente ohne Delimiter (<i>yyyymmdd</i>).
DF=L	10-Byte-Darstellung mit 4-stelliger Jahreskomponente und Delimitern (<i>yyyy-mm-dd</i>).

Anmerkung:

Die Reihenfolge der Tages-, Monats- und Jahreskomponenten sowie die Delimiterzeichen werden durch den DTFORM-Profilparameter (siehe Natural Referenz-Dokumentation) bestimmt.

Bei DF=S stehen nur 2 Stellen für die Jahresinformation zur Verfügung; selbst wenn der Datumswert das Jahrhundert enthielte, würden folglich diese Informationen bei der Umsetzung verloren gehen.

Mit DF=I bzw. DF=L können Sie Ihre Anwendungen nach und nach auf 4-stellige Jahresdarstellung umstellen und dabei weiterhin die durch den DTFORM-Parameter gebotene Flexibilität ausnutzen.

Der DF-Parameter wird bei der Kompilierung ausgewertet. Er kann angegeben werden in einem FORMAT-Statement, den Statements INPUT, DISPLAY, WRITE und PRINT (auf Statement- und Feldebene) sowie den Statements MOVE, COMPRESS, STACK, RUN und FETCH (auf Feldebene).

Siehe auch Abschnitt **Verarbeitung von Datumsinformationen** im *Natural Leitfaden zur Programmierung*.

DFOUT — Datumsformat für Ausgabe

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
DFOUT	S, I	S	SET GLOBALS

Dieser Session-Parameter bestimmt die Form, in der die Werte von Datumsvariablen mit INPUT-, DISPLAY-, PRINT- und WRITE-Statements angezeigt werden.

DFOUT=S	Datumsvariablen werden mit 2-stelliger Jahreskomponente und mit durch den Profilparameter DTFORM bestimmten Delimiterzeichen angezeigt. Zum Beispiel: <i>yy-mm-dd</i> .
DFOUT=I	Datumsvariablen werden mit vollständiger 4-stelliger Jahreskomponente und ohne Delimiterzeichen angezeigt. Zum Beispiel: <i>yyyymmdd</i> .

Der DFOUT-Parameter wird zur Laufzeit ausgewertet. Er gilt für Datumsfelder in INPUT-, DISPLAY-, PRINT- und WRITE-Statements, für die weder explizit Editiermasken angegeben sind noch der Session-Parameter DF (siehe Seite 116) gesetzt ist.

Bei beiden DFOUT-Einstellungen wird die Reihenfolge der Tages-, Monats- und Jahreskomponenten in den Datumswerten durch den Profilparameter DTFORM (siehe *Natural Referenz-Dokumentation*) bestimmt.

Siehe auch Abschnitt **Verarbeitung von Datumsinformationen** im *Natural Leitfaden zur Programmierung*.

DFSTACK — Datumsformat für Stack

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
DFSTACK	S, C, I	S	SET GLOBALS

Dieser Session-Parameter bestimmt die Form, in der die Werte von Datumsvariablen mit einem STACK-, RUN- oder FETCH-Statement auf dem Stack abgelegt werden.

DFSTACK=S	Datumsvariablen werden mit 2-stelliger Jahreskomponente und mit durch den Profilparameter DTFORM bestimmten Delimiterzeichen auf dem Stack abgelegt. Zum Beispiel: <i>yy-mm-dd</i> .
DFSTACK=C	Wie DFSTACK=S. Außerdem wird eine Änderung des Jahrhunderts (d.h. wenn das Jahrhundert, das genommen wird, wenn der Wert vom Stack gelesen wird, nicht dasselbe ist wie das Jahrhundert des ursprünglichen Datumswertes, bevor er auf dem Stack abgelegt wurde) zur Laufzeit abgefangen.
DFSTACK=I	Datumsvariablen werden mit vollständiger 4-stelliger Jahreskomponente und ohne Delimiterzeichen auf dem Stack abgelegt. Zum Beispiel: <i>yyyymmdd</i> .

Der DFSTACK-Parameter gilt nicht für STACK-, RUN- und FETCH-Statements, für die der Session-Parameter DF (siehe Seite 116) gesetzt ist.

Siehe auch Abschnitt **Verarbeitung von Datumsinformationen** im *Natural Leitfaden zur Programmierung*.

DFTITLE — Datumsformat in Standard-Seitenüberschrift

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
DFTITLE	S, L, I	S	SET GLOBALS

Dieser Session-Parameter bestimmt die Form des Datums in einer Standard-Seitenüberschrift (wie sie mit einem DISPLAY-, WRITE- oder PRINT-Statement ausgegeben wird).

DFTITLE=S	Das Datum wird mit 2-stelliger Jahreskomponente und Delimiterzeichen ausgegeben. Zum Beispiel: <i>yy-mm-dd</i> .
DFTITLE=L	Das Datum wird mit 4-stelliger Jahreskomponente und Delimiterzeichen ausgegeben. Zum Beispiel: <i>yyyy-mm-dd</i> .
DFTITLE=I	Das Datum wird mit 4-stelliger Jahreskomponente ohne Delimiterzeichen ausgegeben. Zum Beispiel: <i>yyyymmdd</i> .

Anmerkung:

Die Reihenfolge der Tages-, Monats- und Jahreskomponenten sowie die Delimiterzeichen werden durch den DTFORM-Profilparameter (siehe Natural Referenz-Dokumentation) bestimmt.

Der DFTITLE-Parameter wird zur Laufzeit ausgewertet. Er hat keine Auswirkungen auf benutzer-definierte Seitenüberschriften (wie sie mit einem WRITE TITLE-Statement angegeben werden).

Siehe auch Abschnitt **Verarbeitung von Datumsinformationen** im *Natural Leitfaden zur Programmierung*.

DU — Dump-Generierung

Dieser Parameter steht für

- Großrechner
- UNIX/OpenVMS und Windows

zur Verfügung.

DU auf Großrechnern

Mit diesem Session-Parameter bestimmen Sie, ob ein Memory-Dump generiert wird, falls die Ausführung eines Natural-Programms aufgrund eines Fehlers abgebrochen wird.

DU=OFF	Bei einem Programmabbruch wird kein Memory-Dump generiert und die Fehlermeldung NAT0954 ABEND at execution time (ABBRUCH zur Ausführungszeit) ausgegeben. Was anschließend geschieht, ist abhängig von der Einstellung des Profilparameters CC.
DU=ON	Bei einem Programmabbruch wird ein Memory-Dump generiert.
DU=SNAP	Bei einem Programmabbruch im Verlauf der Natural-Session wird sofort ein Dump erzeugt. Die Session wird anschließend fortgesetzt.
DU=FORCE	Bei einem Programmabbruch im Verlauf der Natural-Session wird sofort ein Dump erzeugt und die Session sofort abgebrochen. Dies kann in manchen Umgebungen zu Testzwecken sinnvoll sein.

Beispiel:

```
SET GLOBALS DU=ON
```

Anmerkungen:

Seien Sie vorsichtig, wenn Sie diesen Parameter benutzen, weil alle für den Benutzer gerade aktiven Programme und Subroutinen im Natural Buffer Pool zurückgehalten werden.

DU=ON, SNAP bzw. FORCE kann sich aufgrund einer möglichen Puffer-Fragmentierung merklich negativ auf die System-Verarbeitungszeit auswirken.

Anmerkung für UTM:

Unter UTM wird dieser Parameter ignoriert, und im Falle eines Programmabbruchs wird immer ein Dump erzeugt.

DU unter OpenVMS, UNIX und Windows

Mit diesem Session-Parameter bestimmen Sie, ob ein disassemblierter Objektcode-Dump generiert werden soll.

DU=ON	<p>Wenn eines der Systemkommandos CHECK, STOW, CATALOG oder RUN ausgeführt wird, wird eine disassemblierte Objektcode-Datei erzeugt. Diese Dump-Datei wird in das Verzeichnis "tmp" unterhalb des Natural-Hauptverzeichnisses geschrieben.</p> <p>Der Name der Dump-Datei besteht aus dem Namen der Source-Datei und der Erweiterung ".DIA". Wenn die Source-Datei nicht gesichert wurde, ist der Name der Dump-Datei "GEN.DIA" (unter OpenVMS und UNIX) oder "UNTITLED.DIA" (unter Windows).</p> <p>Enthält das Programm Datenbank-Statements, werden außerdem Dump-Dateien mit der Erweiterung ".ADA" (für Adabas) bzw. ".SQL" (für SQL-Datenbanken) erzeugt.</p> <p>Wenn XREF-Daten generiert werden, wird außerdem eine Datei mit der Erweiterung ".XRF" erzeugt.</p>
DU=OFF	Es wird kein Dump erzeugt.

Beispiel:

```
SET GLOBALS DU=ON
```

Anmerkung:

Die mit *DU=ON* erzeugte Dump-Datei kann (je nach Größe der Source-Datei) sehr groß sein, was sich merklich negativ auf die System-Verarbeitungszeit auswirken kann.

DY — Dynamische Attribute

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
DY	siehe unten	keiner	DISPLAY INPUT PRINT WRITE

Mit diesem Session-Parameter werden Anzeigefeldern dynamisch Feldattribute zugewiesen.

Anfang und Ende einer Attribut-Definition werden mit besonderen Identifizierungszeichen (Escape-Zeichen) markiert.

Ein alphanumerisches Feld, das in einem INPUT-, DISPLAY-, WRITE- oder PRINT-Statement verarbeitet wird und Identifizierungszeichen enthält, wird an den Identifizierungszeichen getrennt und in Teilfelder aufgeteilt. Dann wird dem Teilfeld das entsprechende Attribut zugeordnet, und die Identifizierungszeichen werden durch Leerzeichen ersetzt.



- ① Ein Zeichen, das den Anfang der Attributdefinition markiert. Als Zeichen können Sie ein beliebiges Sonderzeichen (*c*) oder eine Hexadezimalzahl mit einem vorangestellten Apostroph (*'xx*) verwenden.
- ② Das zuzuordnende Farbattribut (zur Erklärung der Farbcodes siehe Session-Parameter CD, Seite 111).
- ③ Überschreib-Schutz für das Teilfeld.
- ④ Weitere zuzuordnende Attribute (siehe Session-Parameter AD, Seite 104).

- ⑤ Ein Zeichen, das das Ende der Attributdefinition markiert. Als Zeichen können Sie ein beliebiges Sonderzeichen (*c*) oder eine Hexadezimalzahl mit einem vorangestellten Apostroph (*'xx*) verwenden.

Sie können bis zu sieben Attributsequenzen (Anfangsidentifizierungszeichen und Attribute) vor dem Zeichen, das das Ende der Attributdefinitionen bestimmt, angeben.

Anmerkung:

Ist für einen Teil eines Feldes eine mit dem DY-Parameter gemachte Angabe wirksam, werden mit dem CV-Parameter gemachte Angaben für diesen Feldteil ignoriert.

Beispiel 1:

DY = <U>

Die Textkette: **THIS <is> UNDERLINED**

wird ausgegeben als: **THIS is UNDERLINED**

Beispiel 2:

DY = <BL[RE/GR>

Weist zu: Blau zu "<"

 Rot zu "["

 Grün zu "/"

">" schaltet wieder zur ursprünglichen Farbe des Feldes zurück.

Beispiel 3:

DY = <P>

Die Textkette: **Do not overwrite <this>**

wird ausgegeben als: **Do not overwrite this** (wobei "this" geschützt ist)

EJ — Eject (Seitenvorschub)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
EJ	ON / OFF	ON	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, ob am Ende einer logischen Seite, beim Wechsel zwischen Programmeingabe und -ausgabe und nach der Meldung “normal end” ein Seitenvorschub erfolgen soll.

EJ=ON	Ein Seitenvorschub wird ausgeführt.
EJ=OFF	Ein Seitenvorschub wird <i>nicht</i> ausgeführt. Bei Testläufen, bei denen Seitenumbrüche keine Rolle spielen, empfiehlt es sich, EJ=OFF zu setzen, um Papier zu sparen.

Die Einstellung des EJ-Parameters kann mit dem Statement EJECT überschrieben werden.

Der EJ-Parameter gilt nur für den ersten auszugebenden Report (Report 0). Für weitere Reports ist das EJECT-Statement mit Report-Spezifikation zu verwenden.

Beispiel:

```
SET GLOBALS EJ=OFF
```

EM — Editiermaske

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
EM	siehe unten	keiner	DEFINE DATA DISPLAY FORMAT INPUT MOVE EDITED PRINT WRITE

Mit diesem Session-Parameter können Sie für ein Eingabe- und/oder Ausgabefeld, das in einem DISPLAY-, INPUT-, MOVE EDITED-, PRINT- oder WRITE-Statement verwendet wird, eine Editiermaske definieren.

Bei Eingabefeldern muß ein Wert genau entsprechend der Editiermaske eingegeben werden. Um die Editiermaske für die Eingabe sichtbar zu machen, sollte ein Eingabefeld als modifizierbar (AD=M) definiert werden.

Für ein Datenbankfeld kann im DDM bereits eine Standard-Editiermaske definiert sein. Wenn Sie mit dem EM-Parameter eine Editiermaske für ein Datenbankfeld angeben, so gilt diese anstelle einer möglicherweise im DDM für das Feld definierten Standard-Editiermaske.

Setzen Sie für ein Feld EM=OFF, so wird für das Feld *keine* Editiermaske verwendet, auch keine möglicherweise im DDM definierte.

Auf *Statement-Ebene* eines DISPLAY-, FORMAT-, INPUT- oder WRITE-Statements kann keine Editiermaske definiert, sondern allenfalls EM=OFF gesetzt werden.

Wenn eine Editiermaske definiert ist, so überschreibt diese etwaige Einstellungen der Session-Parameter AL, NL und SG.

Beispiele:

```
DISPLAY AA(EM=OFF) AB(EM=XX.XX)
WRITE SALARY (EM=ZZZ,ZZ9)
```

Lange Editiermasken können in Form einer Kurzschreibweise definiert werden. Die folgenden Beispiele zeigen die für numerische, hexadezimale und alphanumerische Editiermasken verwendbare Kurzschreibweise:

EM=9(4)–9(5) entspricht: **EM=9999–99999**
EM=H(10) entspricht: **EM=HHHHHHHHHH**
EM=X(6)..X(3) entspricht: **EM=XXXXXX..XXX**

Diese Schreibweise gilt nur für die Zeichen “9”, “H”, “X” und “Z”, mit denen bei numerischen, hexadezimalen und alphanumerischen Editiermasken die signifikanten Stellen dargestellt werden.

Leerzeichen in Editiermasken

Leerzeichen innerhalb einer Editiermaske lassen sich durch das Zeichen auf Ihrer Tastatur darstellen, das in Hexadezimalcode H’20’ (ASCII) bzw. H’5F’ (EBCDIC) entspricht, d.h. das Zeichen “^” (oder “-”).

Standard-Editiermasken

Wenn Sie für ein Feld keine Editiermaske angeben, erhält das Feld entsprechend seinem Format eine Standard-Editiermaske:

Feldformat	Standard-Editiermaske
A	X
B	H
N, P, I	Z9
F	wissenschaftliche Darstellung
D	abhängig vom Standard-Datumsformat (wie mit dem Profilparameter DTFORM gesetzt)
T	HH:II:SS
L	Leerzeichen / X

Editiermasken für numerische Felder

Eine für Felder mit Format N, P, I oder F definierte Editiermaske muß mindestens eine "9" oder ein "Z" enthalten. Enthält sie mehr "9" und "Z" als der Feldwert lang ist, wird die Anzahl der Ausgabestellen in der Editiermaske der Anzahl der für den Feldwert definierten Stellen angepaßt. Hat die Editiermaske weniger signifikante Stellen als der Feldwert, wird der Feldwert um die entsprechende Anzahl der Stellen vor bzw. nach dem Komma verkürzt ausgegeben.

Es gibt zwei Optionen bei numerischen Editiermasken. Die erste ermöglicht das Einfügen eines Sonderzeichens als erstes Zeichen einer Editiermaske.

Zeichen zur Definition numerischer Editiermasken:

Zeichen	Bedeutung
9	Auszugebende Stelle (eine Stelle des Feldwertes).
Z	Nullunterdrückung bei vorangestellten Nullen. Dies gilt standardmäßig für numerische Felder. Zur gleitenden Nullunterdrückung kann das "Z" mehrmals angegeben werden. Rechts vom Komma (Dezimalpunkt) darf kein "Z" stehen. Ein Nullwert kann unter Einbeziehung aller "Z" in der Editiermaske als lauter Leerzeichen ausgegeben werden (vgl. Session-Parameter ZP, Seite 184).
+	Ein gleitendes Vorzeichen, das vor/nach der Zahl ausgegeben werden soll. Das Zeichen wird je nach Wert der Zahl als "+" oder "-" generiert.
-	Ein gleitendes Minus-Vorzeichen, das vor/nach der Zahl ausgegeben werden soll, wenn die Zahl negativ ist.
S	Ein Vorzeichen, das vor dem Feld ausgegeben werden soll. Das Vorzeichen wird je nach Wert der Zahl als "+" oder "-" generiert. Wenn "S" verwendet wird, muß es als erstes Zeichen einer Editiermaske angegeben werden.
N	Ein Minus-Vorzeichen, das vor dem Feld ausgegeben werden soll, wenn der Feldwert negativ ist.
.	Ein Punkt, als erstes Zeichen verwendet, stellt ein Komma (Dezimalpunkt) dar und ist signifikant.
H	Kennzeichnet den Anfang einer hexadezimalen Editiermaske. Ist die erste Stelle einer Editiermaske ein "H", werden alle anderen Zeichen, die nicht "H" sind, als Einschubzeichen interpretiert.

Die zweite Option besteht darin, der ersten Stelle, die einen Teil der Zahl enthält (einer mit "9" definierten Stelle), eine beliebige Anzahl von Zeichen voranzustellen. Das erste dieser vorangestellten Zeichen darf keines der in obiger Tabelle aufgeführten Zeichen sein, es sei denn, es wird in Apostrophen angegeben.

Das erste dieser Zeichen wird nur ausgegeben, falls der Feldwert vorangestellte Nullen enthält und die Editiermaske mit "Z" definiert ist (Nullunterdrückung); das Zeichen wird dann für die vorangestellten Nullen anstelle eines Leerzeichens ausgegeben.

Beide Optionen ermöglichen es, Zeichen zwischen den signifikanten Stellen einzufügen bzw. nachzustellen. Ein vorangestelltes, eingefügtes oder nachgestelltes Leerzeichen wird durch (^) dargestellt. Sollen die signifikanten Zeichen (9, H, Z, X) als Einschubzeichen verwendet werden, müssen sie in Apostrophen stehen.

Andere, nichtsignifikante Zeichen brauchen nicht in Apostrophen zu stehen. Innerhalb einer Editiermaske ist es möglich, vorangestellte, eingefügte und nachgestellte Zeichenketten zu haben, von denen einige in Apostrophen stehen und andere nicht.

Ein nachfolgendes Vorzeichen wird durch ein "+" oder "-" als letztes Zeichen der Editiermaske angegeben. Ein "+" bewirkt, daß das Vorzeichen je nach Feldwert entweder als "+" oder "-" ausgegeben wird; ein "-" bewirkt, daß bei einem positiven Feldwert ein Leerzeichen und bei einem negativen Feldwert ein "-" ausgegeben wird. Ist für eine Editiermaske ein vorangestelltes und ein nachgestelltes Vorzeichen definiert, werden beide ausgegeben.

Beispiele für numerische Editiermasken:

Die folgende Tabelle zeigt in der oberen Zeile die Werte numerischer Felder (Format N), wie sie ohne Editiermaske ausgegeben würden, und darunter die unter Verwendung der verschiedenen Editiermasken ausgegebene Form:

Wert	0000.03	-0054	+0087	0962	1830
Format Editiermaske	(N4.2)	(N4)	(N4)	(N4)	(N4)
EM=9.9	0.0	4.	7.	2.	0.
EM=99	00	54	87	62	30
EM=S99	+00	-54	+87	+62	+30
EM=+Z9	+0	-54	+87	+62	+30
EM=-9.99	0.03	-4.	7.	2.	0.
EM=N9	0	-4	7	2	0
EM=*9.99	0.03	4.	7.	2.	0.
EM=Z99	00	54	87	962	830
EM=*DMZZ9.9	DM**0.0	DM*54.	DM*87.	DM962.	DM830.
EM=999+	000+	054-	087+	962+	830+
EM=999-	000	054-	087	962	830
IC=\$ EM=ZZZ.99	\$.03	\$54.	\$87.	\$962.	\$830.
EM=H(6)					
- ASCII:	303030303033	30303574	30303837	30393632	31383330
- EBCDIC:	F0F0F0F0F0F3	F0F0F5D4	F0F0F8F7	F0F9F6F2	F1F8F3F0

Durch Kombination von Editiermasken mit den Parametern IC und TC ist es bei einem DISPLAY-Statement möglich, negative Zahlen in verschiedenen Formen auszugeben.

Editiermasken für alphanumerische Felder

Für mit Format "A" definierte Felder kann eine alphanumerische Editiermaske definiert werden; sie muß mindestens ein "X" enthalten; jedes "X" steht für ein auszugebendes Zeichen. Ein "H" als erstes Zeichen kennzeichnet eine hexadezimale Editiermaske (siehe Seite 133). Ein Leerzeichen wird durch (^) dargestellt.

Alle anderen Zeichen — außer Klammern — können als vorangestellte, eingeschobene oder nachgestellte Zeichen verwendet werden, wobei diese Zeichen wahlweise durch Apostrophe eingegrenzt werden können oder nicht. Sollen die Zeichen "X", "(" oder " (Anführungszeichen) als nichtsignifikante Einschubzeichen verwendet werden, müssen sie in Apostrophen angegeben werden.

Werden dem ersten signifikanten "X" Zeichen vorangestellt, wird das erste dieser Zeichen nicht ausgegeben. Zeichen, die unmittelbar auf das letzte signifikante "X" folgen, werden ausgegeben.

Ist die Editiermaske kürzer als das Feld, wird die Anzahl der ausgegebenen Stellen auf die Länge der Editiermaske gekürzt.

Beispiel für alphanumerische Editiermasken:

Das folgende Programm definiert Editiermasken für ein Feld mit Format/Länge A4, das den Wert "BLUE" enthält:

```
* EXAMPLE 'EMMASK1'
DEFINE DATA LOCAL
1 #TEXT (A4) INIT <'BLUE'>
END-DEFINE
WRITE NOTITLE 'MASK 1:' 5X #TEXT (EM=X.X.X.X)
/ 'MASK 2:' 5X #TEXT (EM=X^X^X^X)
/ 'MASK 3:' 5X #TEXT (EM=X—X—X)
/ 'MASK 4:' 5X #TEXT (EM=X—X—X—X—X)
/ 'MASK 5:' 5X #TEXT (EM=X' 'X' 'X' 'X)
/ 'MASK 6:' 5X #TEXT (EM=XX...XXX)
/ 'MASK 7:' 5X #TEXT (EM=1234XXXX)
END
```

MASK 1:	B.L.U.E
MASK 2:	B L U E
MASK 3:	B—L—U
MASK 4:	B—L—U—E—
MASK 5:	B L U E
MASK 6:	BL...UE
MASK 7:	234BLUE

Hexadezimale Editiermasken

Wird als erstes Zeichen einer Editiermaske ein “H” angegeben, so wird der Wert eines alphanumerischen oder numerischen Feldes in hexadezimaler Form ausgegeben. Jedes “H” steht für zwei Hexadezimalstellen, die jeweils einem numerischen/alphanumerischen Byte entsprechen.

Alle anderen Zeichen können als Einschubzeichen oder nachgestellte Zeichen verwendet werden, wobei sie wahlweise durch Apostrophe eingegrenzt werden können oder nicht. Ist die Editiermaske kürzer als das Feld, wird der Feldwert entsprechend verkürzt ausgegeben. Ist das Feld kürzer als die Editiermaske, wird die Editiermaske der Feldlänge entsprechend verkürzt ausgegeben.

Alle mit einer hexadezimalen Editiermaske angezeigten Felder werden als alphanumerische Felder behandelt. Ist die Editiermaske kürzer als das Feld, werden daher alle numerischen oder alphanumerischen Stellen von links nach rechts ohne Berücksichtigung von Dezimalstellen ausgegeben.

Editiermasken-Beispiele für hexadezimale Felder:

Die folgenden Tabellen zeigen in der oberen Zeile die Werte eines alphanumerischen Feldes (Format/Länge A2) und dreier numerischer Felder (Format/Länge N2) und darunter die Form, in der sie unter Verwendung der verschiedenen hexadezimalen Editiermasken ausgegeben würden.

ASCII:

Wert →	AB	-10	+10	01
EM=HH	4142	3170	3130	3031
EM=H^H	41 42	31 70	31 30	30 31
EM=HH^H	4142	3170	3130	3031
EM=H-H	41-42	31-70	31-30	30-31
EM=H	41	31	31	30

Anmerkung:

Im Falle von em=h(n) (Hexadezimale Ausgabe) muss der Wert von n im Bereich von 1 <= n <= 126 sein. Ist n > 126, dann wird eine Fehlermeldung angezeigt, wenn die Variable mehr als 126 Elemente hat.

Beispiel:

```
A (A100) := 'A'      /* 100 Zeichen/Elemente
PRINT A (EM=H(200)) /* kein Fehler, da 100<=126, auch wenn 200>126
END
```

EBCDIC:

Wert →	AB	-10	+10	01
EM=HH	C1C2	F1D0	F1F0	F0F1
EM=H:H	C1 C2	F1 D0	F1 F0	F0 F1
EM=HH:H	C1C2	F1D0	F1F0	F0F1
EM=H-H	C1-C2	F1-D0	F1-F0	F0-F1
EM=H	C1	F1	F1	F0

Editiermasken für Datums- und Zeitfelder (Formate D und T)

Zur Definition von Editiermasken für Felder, die mit dem Format D (Datumsfeld) oder T (= Time; Zeitfeld) definiert sind, können folgende Zeichen verwendet werden:

Für Datums- und Zeitfelder (Formate D und T):

Zeichen	Bedeutung
DD	Tag (day).
ZD	Tag mit Nullwertunterdrückung.
MM	Monat.
ZM	Monat mit Nullwertunterdrückung.
YYYY	Jahr (year), vierstellig.
YY	Jahr, zweistellig. (Bei Eingabefeldern wird das aktuelle Jahrhundert vor den Eingabewert gestellt.)
Y	Jahr, einstellig. Darf nicht für Eingabefelder verwendet werden.
WW	Woche.
ZW	Woche mit Nullwertunterdrückung.
JJJ	Julianischer Tag.
ZZJ	Julianischer Tag mit Nullwertunterdrückung.
NN... oder N(<i>n</i>)	Name des Wochentages (sprachabhängig). Die Maximallänge wird durch die Anzahl der N's bzw. durch <i>n</i> bestimmt. Ist der Name länger als die Maximallänge, wird er abgeschnitten; ist er kürzer, wird seine tatsächliche Länge genommen.
O	Nummer des Wochentags (Montag = 1, Dienstag = 2, usw. Ob Montag oder Sonntag als erster Wochentag genommen wird, hängt vom Profilparameter DTFORM ab, wie in Ihrer <i>Natural Referenz-Dokumentation</i> beschrieben).
LL... oder L(<i>n</i>)	Name des Monats (sprachabhängig). Die Maximallänge wird durch die Anzahl der L's bzw. durch <i>n</i> bestimmt. Ist der Name länger als die Maximallänge, wird er abgeschnitten; ist er kürzer, wird seine tatsächliche Länge genommen.
R	Jahr in römischen Ziffern (maximal 13 Stellen).

Wird in einer Eingabe-Editiermaske nur das Jahr (YY oder YYYY) angegeben, aber nicht Monat und Tag, werden die Werte für Monat und Tag jeweils auf "01" gesetzt. Werden in einer Eingabe-Editiermaske nur Jahr (YY oder YYYY) und Monat (MM) angegeben, aber kein Tag, wird der Wert für Tag auf "01" gesetzt.

Wird eine Woche des vorangegangenen oder nachfolgenden Jahres angezeigt, so wird die Anzeige für Jahr entsprechend angepaßt.

Ob eine Woche die 53. Woche des alten oder die 1. Woche des neuen Jahres ist, hängt davon ab, zu welchem Jahr der Donnerstag dieser Woche gehört: liegt er im alten Jahr, gehört die Woche zum alten Jahr; liegt er im neuen Jahr, gehört die Woche zum neuen Jahr.

Anmerkung:

"MM" (bzw. "ZM") und "LL" (bzw. "L(n)") dürfen nicht zusammen in einer Editiermaske angegeben werden.

"NN" (bzw. "N(n)") und "O" dürfen nicht zusammen in einer Editiermaske angegeben werden.

Nur für Zeitfelder (Format T):

Zeichen	Bedeutung
T	Zehntelsekunden (tenths of a second).
SS	Sekunden.
ZS	Sekunden mit Nullwertunterdrückung.
II	Minuten.
ZI	Minuten mit Nullwertunterdrückung.
HH	Stunden.
ZH	Stunden mit Nullwertunterdrückung.
AP	AM/PM-Element (englische Zeitangabe: AM = vormittags, PM = nachmittags).

Beispiele für Datums- und Zeit-Editiermasken:

```

* EXAMPLE 'EMDATI': EDIT MASKS FOR DATE AND TIME
*****
WRITE NOTITLE
  'DATE INTERNAL : ' *DATX (DF=L) /
  '                : ' *DATX (EM=N(9)' 'ZW.'WEEK 'YY) /
  '                : ' *DATX (EM=ZZJ' .DAY 'YYYY)      /
  '    ROMAN       : ' *DATX (EM=R) /
  '    AMERICAN    : ' *DATX (EM=MM/DD/YYYY)          12X 'OR ' *DAT4U /
  '    JULIAN      : ' *DATX (EM=YYYYJJJ)            15X 'OR ' *DAT4J /
  '    GREGORIAN: ' *DATX (EM=ZD.' 'L(10)' 'YYYY) 5X 'OR ' *DATG ///
*
  'TIME INTERNAL : ' *TIMX                          14X 'OR ' *TIME /
  '                : ' *TIMX (EM=HH.II.SS.T) /
  '                : ' *TIMX (EM=HH.II.SS' 'AP) /
  '                : ' *TIMX (EM=HH)
END

```

```

DATE INTERNAL : 1999-01-19
                : Tuesday 3.WEEK 1999
                : 19.DAY 1999
    ROMAN       : MCMXCIX
    AMERICAN    : 01/19/1999          OR    01/19/1999
    JULIAN      : 1999019             OR    1999019
    GREGORIAN: 19.January 1999       OR    19 January 1999

TIME INTERNAL : 13:40:59              OR    13:40:59.5
                : 13.40.59.5
                : 01.40.59 PM
                : 13

```

Editiermasken für logische Felder (Format L)

Editiermasken für Felder, die das Format L haben (logische Felder), können wie folgt definiert werden, wobei *false-string* für die für “falsch” auszugebende Zeichenkette und *true-string* für die für “wahr” auszugebende Zeichenkette steht:

(EM=[*false-string*]/*true-string*)

Die *false-string* darf nicht länger als 31 Zeichen sein.

Beispiel für Editiermasken für logisches Feld:

```

/* EXAMPLE 'EMLOGV'
/* EXAMPLE OF LOGICAL VARIABLE IN LOGICAL CONDITION
/*****
DEFINE DATA LOCAL
1 #SWITCH (L) INIT <TRUE>
1 #INDEX (I1)
END-DEFINE
/*****
FOR #INDEX 1 5
  WRITE NOTITLE #SWITCH (EM=FALSE/TRUE) 5X 'INDEX =' #INDEX
  WRITE NOTITLE #SWITCH (EM=OFF/ON) 7X 'INDEX =' #INDEX
  IF #SWITCH
    MOVE FALSE TO #SWITCH
  ELSE
    MOVE TRUE TO #SWITCH
  END-IF
/*****
SKIP 1
END-FOR
END

```

TRUE	INDEX =	1
ON	INDEX =	1
FALSE	INDEX =	2
OFF	INDEX =	2
TRUE	INDEX =	3
ON	INDEX =	3
FALSE	INDEX =	4
OFF	INDEX =	4
TRUE	INDEX =	5
ON	INDEX =	5

ENDIAN — Endian-Modus für kompilierte Objekte

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
ENDIAN	DEFAULT/LITTLE/BIG	DEFAULT	Systemkommando GLOBALS

Dieser Session-Parameter steht nur für UNIX/OpenVMS und Windows zur Verfügung.

Mit diesem Session-Parameter wird die Architektur angegeben, für die der Compiler von Natural generierte Programme (GP) erzeugen sollte. Einzelheiten siehe Profilparameter ENDIAN.

Beispiel:

```
GLOBALS ENDIAN=BIG
```

ES — Empty Line Suppression (Leerzeilenunterdrückung)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
ES	ON / OFF	OFF	DISPLAY FORMAT WRITE

Mit diesem Session-Parameter können Sie unterdrücken, daß von einem DISPLAY- oder WRITE-Statement erzeugte Leerzeilen ausgegeben werden.

Wenn Sie ES=ON setzen, wird eine von einem DISPLAY- oder WRITE-Statement erzeugte Zeile, die nur Leerwerte enthält, nicht ausgegeben. Dies ist besonders sinnvoll bei der Ausgabe von Arrays (z.B. multiplen Feldern oder Feldern einer Periodengruppe), um nicht überflüssigerweise viele Leerzeilen auszudrucken bzw. anzuzeigen.

Um die Leerwertunterdrückung auch für numerische Werte zu erhalten, muß für die betreffenden Felder neben ES=ON auch der Parameter ZP=OFF gesetzt werden, was bewirkt, daß Nullwerte in Leerwerte umgesetzt und dann ebenfalls nicht ausgegeben werden.

Vgl. Session-Parameter IS (Seite 156) und ZP (Seite 184).

Beispiel:

```
DISPLAY (ES=ON) NAME CITY
```

FC — Filler Character (Füllzeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
FC	jedes Zeichen	Leerzeichen	DISPLAY FORMAT

Mit diesem Session-Parameter bestimmen Sie das Füllzeichen, mit dem bei Spaltenüberschriften, die über ein DISPLAY-Statement erzeugt werden, der Platz rechts und links der Überschrift aufgefüllt wird. Dies gilt nur, wenn die Spaltenbreite von der Feldlänge und nicht von der Länge der Überschrift bestimmt wird (vgl. Session-Parameter HW, Seite 150); andernfalls wird der FC-Parameter ignoriert.

Beispiel:

```
DISPLAY (FC=*)
```

FCDP — Filler Character for Dynamically Protected Fields (Füllzeichen für dynamisch geschützte Felder)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
FCDP	ON / OFF	ON	SET GLOBALS

Mit diesem Session-Parameter können Sie die Anzeige von Füllzeichen für Eingabefelder unterdrücken, die dynamisch schreibgeschützt wurden (d.h. denen das Attribut AD=P mittels einer Kontrollvariable zugewiesen wurde).

Je nach dem Wert dieses Parameters werden dynamisch geschützte Eingabefelder entweder mit Leerzeichen oder mit den definierten Füllzeichen angezeigt:

FCDP=ON	Dynamisch geschützte Eingabefelder werden mit Füllzeichen gefüllt angezeigt. Dies kann bei Benutzern den Eindruck erwecken, sie könnten in diese Felder etwas eingeben.
FCDP=OFF	Dynamisch geschützte Eingabefelder werden mit Leerzeichen gefüllt angezeigt.

Beispiel:

```
DEFINE DATA LOCAL
1 #FIELD1 (A5)
1 #FIELD2 (A5)
1 #CVAR1 (C) INIT <(AD=P)>
1 #CVAR2 (C)
END-DEFINE
*
INPUT #FIELD1 (AD=Y'_' CV=#CVAR1) /* Feld ist schreibgeschützt
      #FIELD2 (AD=Y'_' CV=#CVAR2) /* Feld ist nicht schreibgeschützt
...
END
```

Anzeige mit FCDP=ON: #FIELD1 _____ #FIELD2 _____

Anzeige mit FCDP=OFF: #FIELD1 #FIELD2 _____

FL — Floating Point Mantissa Length Fließpunkt-Mantissenlänge

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
FL	1 bis 16	keiner	DISPLAY FORMAT INPUT PRINT WRITE

Mit diesem Session-Parameter bestimmen Sie die Mantissen-Länge einer Fließpunkt-Variablen (Gleitkomma-Variablen) während der Ein- oder Ausgabe.

Die Gesamtlänge für Vorzeichen, Exponent und Dezimalkomma ist “FL + 6”.

Beispiel:

DISPLAY FL=5 → +1.2345E+03

FS — Format-Spezifikation

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
FS	ON / OFF	OFF	SET GLOBALS

Dieser Parameter gilt nur im Reporting Mode.

Mit diesem Session-Parameter bestimmen Sie, ob für die Definition von Benutzervariablen Standardformat und -länge gelten sollen.

FS=OFF	Eine Benutzervariable in einem Natural-Programm, für die Format und Länge nicht explizit definiert sind, erhält Standardformat/-länge N7.
FS=ON	Einer neuen Benutzervariablen wird im <i>Reporting Mode</i> von Natural kein/e Standardformat/-länge zugeordnet; Sie müssen Format und Länge explizit definieren.

Beispiel:

```
SET GLOBALS FS=ON
```

GC — Filler Character for Group Headers (Füllzeichen für Gruppenüberschriften)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
GC	jedes Zeichen	Leerzeichen	DISPLAY FORMAT

Mit diesem Session-Parameter definieren Sie das Füllzeichen, mit dem bei Überschriften, die über ein DISPLAY-Statement erzeugt werden, der Platz rechts und links der Überschrift aufgefüllt wird. Im Gegensatz zum Session-Parameter FC (Seite 142) gilt dieses Zeichen für Überschriften, die über mehrere Spalten, also über eine Gruppe von Feldern, gehen.

Beispiel:

```
DISPLAY (GC=*)
```

HC — Header Centering (Überschriften-Zentrierung)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
HC	C, L, R	C	DISPLAY FORMAT

Mit diesem Session-Parameter bestimmen Sie die Ausrichtung von Spaltenüberschriften.

HC=C	Spaltenüberschriften werden zentriert ausgegeben.
HC=L	Spaltenüberschriften werden linksbündig ausgegeben.
HC=R	Spaltenüberschriften werden rechtsbündig ausgegeben.

Beispiel:

```
DISPLAY (HC=L)
```

HE — Helproutine

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
HE	siehe unten	keiner	INPUT

Mit diesem Session-Parameter können Sie den Namen einer Helproutine oder Map angeben, die einem Feld zugewiesen werden soll.

Helproutinen können mit dem Natural-Programm-Editor erstellt werden, Help Maps mit dem Natural-Map-Editor.

Die zugewiesene Helproutine kann dann bei der Verarbeitung des betreffenden INPUT-Statements oder der betreffenden Map aufgerufen werden, indem der Benutzer ein Fragezeichen “?” (bzw. ein anderes als Hilfe-Zeichen definiertes Zeichen) in das Feld eingibt oder den Cursor in das Feld plaziert und die mit dem SET KEY-Statement definierte Hilfe-Funktionstaste drückt.

Der Parameter verwendet die folgende Syntax:

$$HE=operand1 \left[, \left[\begin{array}{l} operand2 \\ = \end{array} \right] \right] \dots_{20}$$

Operand	Mögliche Struktur	Mögliche Formate	Referenzierung erlaubt	Dynam. Definition
Operand1	C S	A	nein	nein
Operand2	C S A	A N P I F B D T L C G O	nein	nein

Operand1 ist der Name der Helproutine oder Map, die aufgerufen werden soll. Der Name kann eine 1 bis 8 Zeichen umfassende alphanumerische Konstante oder Benutzervariable sein. Wenn eine Variable verwendet wird, muss sie vorher definiert worden sein. Der Name kann ein Kaufmännisches Und (&) enthalten; zur Ausführungszeit wird dieses Zeichen durch den aktuellen Wert der Natural-Systemvariablen *LANGUAGE ersetzt. Diese Funktion ermöglicht die Verwendung von mehrsprachigen Helproutinen oder Maps.

Nach dem Helproutinen- oder Map-Namen können Sie 1 bis 20 Parameter (*Operand2*) angeben, die an die Helproutine oder Map übergeben werden. Diese können Konstanten oder Benutzervariablen sein, die die Parameterwerte enthalten. Geben Sie “=” als Parameter an, wird der Name des Feldes (unter dem es in der Map definiert ist) an die Helproutine übergeben. Ist die Helproutine nicht einem Feld, sondern einer Map zugeordnet, wird mit “=” der Map-Name übergeben.

Anmerkung:

Die Operanden müssen entweder mit dem Input-Delimiterzeichen (wie mit dem Session-Parameter ID definiert) oder mit einem Komma voneinander getrennt werden. Ein Komma darf hierzu allerdings nicht verwendet werden, falls das Komma als Dezimalkomma (mit dem Session-Parameter DC) definiert ist.

Wenn Parameter angegeben werden, so muß die Helproutine mit einem DEFINE DATA PARAMETER-Statement beginnen, in dem Felder definiert werden, die in Format und Länge den übergebenen Parametern entsprechen.

Wird mit “=” ein Feld- bzw. Map-Name übergeben, so *muß* das entsprechende Feld in der Helproutine mit Format/Länge A65 definiert werden.

Der Wert des Feldes, dem die Helproutine zugeordnet ist, kann in der Helproutine referenziert werden. Hierzu muß im DEFINE DATA PARAMETER-Statement der Helproutine ein Feld definiert werden, das in Format und Länge dem ursprünglichen Feld entspricht. Werden in dem DEFINE DATA PARAMETER-Statement noch andere Felder definiert, so muß dieses Feld immer als letztes definiert werden.

Ist das Feld, für das die Helproutine angegeben wird, ein Element eines Arrays, so können die Ausprägungen dieses Feldes von der Helproutine referenziert werden; hierzu müssen Sie Indexparameter mit Format/Länge I2 am Schluß des DEFINE DATA PARAMETER-Statements definieren. Entsprechend der Array-Dimensionen können Sie bis zu drei Indexparameter angeben.

Ausführung von Helproutinen

Wenn eine Helproutine — durch Eingabe eines “?”, Drücken der (mit einem SET KEY-Statement definierten) Hilfetaste oder über ein REINPUT USING HELP-Statement — aufgerufen wird, werden alle in andere Felder eingegebenen Werte erst verarbeitet, nachdem die Ausführung der Helproutine beendet ist.

Anmerkung:

Pro INPUT-Statement ist jeweils nur eine Hilfe-Anforderung möglich. Wenn für mehrere Felder gleichzeitig Hilfe angefordert wird (z.B. durch Eingabe von Fragezeichen in mehrere Felder), wird nur die erste Hilfe-Anforderung ausgeführt.

Beispiel:

```
/* MAIN PROGRAM
DEFINE DATA
1 #A(A20/1:3)
END-DEFINE
...
SET KEY PF1=HELP
...
INPUT #A (2) (HE='HELPA',=)
...
END
```

```
/* HELP-ROUTINE 'HELPA'
DEFINE DATA PARAMETER
1 #VARNAME (A65)
1 #PARM1 (A20)
1 #VARINDEX (I2)
END-DEFINE
...

```

Soll ein Wert von der Helproutine an ein Eingabefeld übergeben werden, muß das Feld als modifizierbar (AD=M) definiert werden.

HW — Heading Width (Überschriftenbreite)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
HW	ON / OFF	ON	DISPLAY FORMAT

Mit diesem Session-Parameter bestimmen Sie die Breite einer mit einem DISPLAY-Statement erzeugten Ausgabespalte.

HW=ON	Die Breite einer DISPLAY-Spalte wird entweder von der Länge der Spaltenüberschrift oder von der Länge des Feldes bestimmt, je nachdem was länger ist. Dies gilt auch, wenn keine Spaltenüberschrift ausgegeben wird, weil entweder das DISPLAY-Statement die NOHDR-Option enthält oder es sich um ein sekundäres DISPLAY-Statement handelt (vgl. DISPLAY-Statement).
HW=OFF	Die Breite einer DISPLAY-Spalte wird allein von der Länge des Feldes bestimmt. HW=OFF gilt nur für DISPLAY-Statements, die keine Spaltenüberschriften erzeugen (d.h. entweder ein erstes DISPLAY-Statement mit NOHDR-Option oder aber auch ein sekundäres DISPLAY-Statement).

Beispiel:

DISPLAY (HW=OFF)

IA — INPUT Assign Character (INPUT-Zuweisungszeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
IA	jedes Sonderzeichen	=	SET GLOBALS

Das mit diesem Session-Parameter definierte Zeichen gilt als Zuweisungszeichen für Eingabe-Parameter bei der Verarbeitung von INPUT-Statements im Keyword/Delimiter-Modus oder bei der Verarbeitung von Daten aus dem Natural-Stack.

Das mit dem IA-Parameter definierte Zeichen muß ein anderes sein als das mit dem CF-Parameter (Steuerzeichen für Terminalkommandos), DC-Parameter (Dezimalkomma) oder ID-Parameter (INPUT-Delimiterzeichen) definierte.

Anmerkung:

Das mit dem IA-Parameter definierte Zeichen sollte ein anderes sein als das mit dem HI-Parameter (Hilfe-Zeichen) definierte.

Beispiel:

```
* Program 'SAMPLE'
DEFINE DATA LOCAL
1 #A (A1)
1 #B (A1)
END-DEFINE
INPUT #A #B
WRITE #A #B
END
```

Geben Sie folgendes Systemkommando ein: **GLOBALS IM=D**

Dann geben Sie folgendes Systemkommando ein: **SAMPLE #A=Y,#B=X**

Das Programm erzeugt folgende Ausgabe: Y X

Geben Sie folgendes Systemkommando ein: **GLOBALS IA=:**

Dann geben Sie folgendes Systemkommando ein: **SAMPLE #B:X,#A:Y**

Das Programm erzeugt folgende Ausgabe: Y X

IC — Insertion Character (Einfügungszeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
IC	jedes Zeichen	keiner	DISPLAY FORMAT

Die mit diesem Session-Parameter angegebene Zeichenkette wird bei einem Feld, das über ein DISPLAY-Statement ausgegeben wird, unmittelbar vor dem Feldwert ausgegeben. Die Ausgabelänge des Feldes vergrößert sich dadurch entsprechend. Sie können eine Zeichenkette von 1 bis 10 Zeichen angeben.

Bei numerischen Werten werden die Einfügungszeichen vor der ersten signifikanten Stelle ausgegeben.

Sie können die Zeichenkette wahlweise in Apostrophen angeben; in diesem Fall darf die Zeichenkette jedes beliebige Zeichen enthalten. Eine Zeichenkette, die Anführungszeichen oder ein schließendes Klammerzeichen enthält, muß in Apostrophen stehen. Ein Leerzeichen in einer nicht durch Apostrophe eingegrenzten Zeichenkette wird durch das Zeichen “^” dargestellt.

Die Parameter IC und LC schließen einander aus.

Beispiele:

```
DISPLAY AA(IC=*)
```

```
DISPLAY SALARY(IC=' $ ')
```

ID — INPUT-Delimiterzeichen

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
ID	jedes Sonderzeichen, Leerzeichen	, (Komma)	SET GLOBALS

Das mit diesem Parameter definierte Zeichen gilt als Delimiterzeichen (Begrenzungszeichen) zum Abgrenzen von Werten bei INPUT-Statements im Keyword/Delimiter-Modus.

Falls das Begrenzungszeichen das Komma sein soll, muß dies als dynamischer Parameter in Apostrophen ',' angegeben werden, da bei dynamischen Parametern das Komma als Begrenzungszeichen der Eingabeparameter gilt.

Das mit dem ID-Parameter definierte Zeichen muß ein anderes sein als das mit dem DC-Parameter (Dezimalkomma) oder IA-Parameter (INPUT-Zuweisungszeichen) definierte.

Anmerkung:

Das mit dem ID-Parameter definierte Zeichen sollte ein anderes sein als das mit dem CF-Parameter (Steuerzeichen für Terminalkommandos) oder HI-Parameter (Hilfe-Zeichen) definierte.

Der Punkt (.) sollte nicht als INPUT-Delimiterzeichen verwendet werden, da dies zu Situationen führen kann, in denen ein Programmende-Punkt fälschlicherweise als INPUT-Delimiterzeichen interpretiert wird.

Beispiele:

```
SET GLOBALS ID=
```

```
PARAM='... ,ID=',' ,...'
```

IM — INPUT-Modus

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
IM	F, D	F	SET GLOBALS

Mit diesem Session-Parameter setzen Sie den INPUT-Modus.

IM=D	Delimiter-Modus.
IM=F	Forms-Modus.

Der Standard-Modus für Video-Terminals ist Forms-Modus.

Der INPUT-Modus kann auch mit den Terminalkommandos `%D` und `%F` geändert werden.

Weitere Informationen siehe INPUT-Statement.

Beispiel:

```
SET GLOBALS IM=D
```

IP — INPUT Prompt (Eingabeaufforderungstext)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
IP	ON / OFF	ON	FORMAT INPUT

Mit diesem Session-Parameter wird bei INPUT-Statements der Text, der zur Eingabe auffordert, gesteuert.

IP=ON	Eingabe-/Ausgabefeldern eines INPUT-Statements, denen kein Textelement vorangestellt ist, wird der betreffende Feldname vorangestellt.
IP=OFF	Es wird kein Eingabeaufforderungstext in Form von Feldnamen generiert; nur wenn einem Feld explizit ein Textelement vorangestellt ist, wird dieser Text als Eingabeaufforderung ausgegeben.

Beispiel:

```
FORMAT IP=OFF
```

IS — Identical Suppress (Unterdrücken identischer Werte)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
IS	ON / OFF	OFF	DISPLAY FORMAT WRITE

Mit diesem Session-Parameter können Sie die mehrfache Ausgabe identischer Feldwerte bei einem WRITE- oder DISPLAY-Statement unterdrücken.

Wenn IS=ON gesetzt ist, wird der Wert eines Feldes nicht angezeigt, falls er mit dem vorherigen Wert des Feldes identisch ist.

Erzeugt ein DISPLAY- oder WRITE-Statement unter Verwendung der VERT-Option oder der Schrägstrich-Notation mehrzeilige Ausgaben, so gilt IS=ON nur für die jeweils erste Zeile.

Mit dem Statement SUSPEND IDENTICAL SUPPRESS können Sie die Wirkung von IS=ON für einen einzelnen Datensatz unterdrücken.

Der IS-Parameter kann in Verbindung mit den Parametern ES und ZP zur Unterdrückung der Ausgabe von Leerzeilen eingesetzt werden.

Beispiel:

```
FORMAT IS=ON
```

KD — Key Definition (PF-Tasten-Anzeige)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
KD	ON / OFF	OFF	FORMAT

Dieser Session-Parameter dient dazu, die PF-Tasten zugewiesenen Namen (siehe SET KEY-Statement) anzeigen zu lassen.

Wenn KD=ON gesetzt ist, wird diese Information automatisch bei jeder mit INPUT, WRITE, DISPLAY oder PRINT erzeugten Ausgabe am unteren Bildschirmrand angezeigt.

Da diese Anzeige zwei Zeilen in Anspruch nimmt, muß die logische Seitenlänge (siehe Session-Parameter PS, Seite 174) entsprechend um zwei Zeilen reduziert werden.

Anmerkung für graphische Benutzeroberflächen:

Falls PF-Tasten definiert sind, werden sie immer angezeigt, unabhängig von der Einstellung dieses Parameters. Falls keine PF-Tasten definiert sind, kann dieser Parameter dazu verwendet werden, die Anzeige der EINGABE-Taste ein- bzw. auszuschalten.

Beispiel:

```
FORMAT KD=ON
```

LC — Leading Characters (Vorangestellte Zeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
LC	jedes Zeichen	keiner	DISPLAY FORMAT

Die mit diesem Session-Parameter angegebene Zeichenkette wird bei einem Feld, das über ein DISPLAY-Statement ausgegeben wird, unmittelbar vor dem Feld ausgegeben. Die Breite der Ausgabespalte vergrößert sich dadurch entsprechend.

Sie können eine Zeichenkette von 1 bis 10 Zeichen definieren. Sie können die Zeichenkette wahlweise in Apostrophen angeben; in diesem Fall darf die Zeichenkette jedes beliebige Zeichen enthalten. Eine Zeichenkette, die Anführungszeichen oder ein schließendes Klammerzeichen enthält, muß in Apostrophen stehen. Ein Leerzeichen in einer nicht durch Apostrophe eingegrenzten Zeichenkette wird durch das Zeichen “^” dargestellt.

Die Session-Parameter LC und IC schließen einander aus.

Beispiel:

```
DISPLAY LC=*
```

LE — Limit Error Processing (Limit-Überschreitung)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
LE	ON / OFF	OFF	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, was geschehen soll, wenn bei der Ausführung einer Verarbeitungsschleife das angegebene Limit (d.h. die maximale Anzahl der Schleifendurchläufe) erreicht wird.

Das Limit kann entweder ein globales oder ein schleifenspezifisch festgesetztes Limit sein (siehe Session-Parameter LT).

LE=OFF	Das Natural-Programm wird normal beendet, sobald das Limit erreicht wird.
LE=ON	Das Natural-Programm wird mit einer entsprechenden Fehlermeldung abgebrochen, sobald das Limit erreicht wird. LE=ON gilt nur für Programme, die von einer in der Systemdatei abgelegten Bibliothek geladen werden, d.h. der Bibliothek SYSTEM, oder einer Bibliothek mit einem Namen, der nicht mit dem Präfix "SYS" anfängt.

Beispiel:

```
SET GLOBALS LE=OFF
```

LS — Line Size (Zeilenlänge)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
LS	2 bis 250 (falls ein Standardseitentitel verwendet wird, ist der Mindestwert 8)	physische Zeilenlänge	DISPLAY FORMAT INPUT SET GLOBALS WRITE

Mit diesem Session-Parameter bestimmen Sie, wieviele Stellen eine von einem DISPLAY-, INPUT- oder WRITE-Statement erzeugte Zeile höchstens lang sein darf.

Beispiel:

```
DISPLAY LS=100 'test'
```

LT — Limit of Records Read Limit für Verarbeitungsschleifen

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
LT	0 bis n (n = Wert des <i>Profil</i> parameters LT bei Session-Start)	99999999	SET GLOBALS

Mit diesem Session-Parameter können Sie das allgemeine Limit für Verarbeitungsschleifen in Natural-Programmen bestimmen, d.h. wieviele Datensätze eine Verarbeitungsschleife maximal verarbeiten darf.

Dieses Limit gilt für jede Verarbeitungsschleife, die mit einem der Statements READ, FIND oder HISTOGRAM ausgelöst wird. Hierbei werden alle gelesenen Datensätze mitgezählt, auch solche, die aufgrund einer WHERE-Klausel zurückgewiesen und nicht weiterverarbeitet werden.

Wenn für eine einzelne Verarbeitungsschleife (mittels eines LIMIT-Statements oder einer Limit-Notation) ein höheres Limit gesetzt ist als mit dem LT-Parameter, gilt das mit dem LT-Parameter gesetzte Limit.

Siehe auch Session-Parameter LE (Seite 159).

Beispiel:

```
SET GLOBALS LT=100
```

MC — Multiple-Value Field Count (Anzahl multipler Feldwerte)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
MC	1 bis 191	1	DISPLAY FORMAT INPUT PRINT WRITE

Dieser Parameter darf nur im Reporting Mode verwendet werden.

Mit diesem Session-Parameter geben Sie an, wieviele Werte eines multiplen Feldes standardmäßig ausgegeben werden sollen, wenn das Feld ohne Index in einem DISPLAY- oder WRITE-Statement angegeben ist.

Beispiel:

```
FORMAT MC=5
```

ML — Msg Line Position (Meldungszeilen-Position)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
ML	T (top = oben) B (bottom = unten)	B	Systemkommando GLOBALS

Mit diesem Session-Parameter geben Sie an, ob die Meldungszeile am oberen (T) oder unteren (B) Bildschirmrand angezeigt wird.

Beispiel:

```
GLOBALS ML=T
```

MP — Maximum Pages (Maximale Seitenzahl)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
MP	1 bis 99999	32767	DISPLAY FORMAT PRINT WRITE

Mit diesem Session-Parameter bestimmen Sie, wieviele Seiten ein Report höchstens erzeugen darf.

Die angegebene Zahl bezieht sich auf die Anzahl der physischen Seiten und ist unabhängig von der Seitennummer der Startseite.

Wird die maximale Seitenzahl überschritten, so wird das Programm mit einer Fehlermeldung abgebrochen.

Beispiel:

```
FORMAT MP=1000
```

MS — Manual Skip (Manuelle Cursor-Positionierung)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
MS	ON / OFF	OFF	FORMAT INPUT

Mit diesem Session-Parameter steuern Sie die Positionierung des Cursors bei der Verarbeitung eines INPUT-Statements.

Ist MS=OFF gesetzt, so wird der Cursor ins nächste Eingabefeld plaziert, sobald der Wert des aktuellen Feldes vollständig eingegeben ist.

Beispiel:

```
INPUT (MS=ON) #A #B
```

Anmerkung:

Die Einstellung MS=ON wird unter BS2000/OSD nicht unterstützt.

MT — Maximum Time (Maximale CPU-Zeit)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
MT	siehe unten	60 (Sekunden)	SET GLOBALS

Dieser Parameter gilt nur für Programme, die auf Großrechnern im Batch-Betrieb ausgeführt werden; bei online ausgeführten Programmen wird das Zeitlimit vom verwendeten TP-Monitor gesteuert.

Mit diesem Session-Parameter bestimmen Sie, wieviel CPU-Zeit ein Natural-Programm in Anspruch nehmen darf. Die Zeit wird in Sekunden angegeben.

MT=0 bedeutet, daß kein CPU-Zeitlimit gesetzt ist.

Die maximale CPU-Zeit ist außerdem vom Betriebssystem abhängig. Überschreitet der mit dem MT-Parameter gesetzte Wert das vom Betriebssystem erlaubte Maximum, wird der Wert entsprechend der Betriebssystem-Vorgaben verringert.

Bei Systemumgebungen, die keine CPU-Zeitmessung unterstützen, wird das Limit als die verstrichene Zeit interpretiert. Bei Systemen ohne Zeitmessung wird das CPU-Zeitlimit ignoriert.

Beispiel:

```
SET GLOBALS MT=5
```

NC — Use of Natural System Commands (Verwendung von Systemkommandos)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
NC	ON / OFF	OFF	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, ob Natural-Systemkommandos während der Natural- Session verwendet werden können.

NC=OFF	Alle Systemkommandos können verwendet werden.
NC=ON	Systemkommandos können nicht verwendet werden — außer FIN, LAST, LOGOFF, LOGON, MAINMENU, RENUMBER, RETURN, SETUP und TECH.

Anmerkung:

Natural-Terminalkommandos und Benutzerkommandos sind von diesem Parameter nicht betroffen.

Beispiel:

GLOBALS NC=OFF

NL — Numerische Länge der Ausgabe

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
NL	siehe unten	keiner	DISPLAY FORMAT INPUT PRINT WRITE

Mit diesem Session-Parameter bestimmen Sie die Standard-Eingabe-/Ausgabelänge eines numerischen Feldes, das in einem DISPLAY-, INPUT-, PRINT- oder WRITE-Statement verwendet wird.

Die Länge wird in der Form *nn.m* angegeben, wobei *nn* für die Stellen vor dem Komma (Dezimalpunkt) und *m* für die Stellen nach dem Komma steht. Die Angabe von Stellen nach dem Komma ist nicht erforderlich. *m* darf nicht größer als 7 sein. Insgesamt darf die Länge von *nn* und *m* zusammen 29 Stellen nicht überschreiten.

Ist die NL-Länge kleiner als die Feldlänge, werden die ausgegebenen Werte entsprechend abgeschnitten, ohne daß dies zu einer Fehlermeldung führt.

Ist die NL-Länge größer als die Feldlänge, werden die freien Stellen mit Leerzeichen aufgefüllt, und es führt zu keinem Fehler, wenn ein Eingabefeld abgeschnitten wird.

Der NL-Parameter darf nicht für Gruppen angegeben werden.

Eine für ein Feld definierte Editiermaske setzt den NL-Parameter für dieses Feld außer Kraft.

Beispiel:

```
DISPLAY #AA(NL=20) #AB(NL=3.2)
```

OPF — Overwriting of Protected Fields by Help routines (Überschreiben geschützter Felder durch Help routines)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
OPF	ON / OFF	ON	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, ob der Inhalt eines schreibgeschützten Feldes (Attribut AD=P) durch eine Help routine, die dem Feld zugewiesen ist, überschrieben werden kann.

OPF=ON	Eine einem Feld zugewiesene Help routine kann den Inhalt des Feldes überschreiben, selbst wenn das Feld schreibgeschützt ist.
OPF=OFF	Die Inhalte schreibgeschützter Felder können nicht durch Help routines überschrieben werden.

Der OPF-Parameter gilt nur für das Feld, für das eine Help routine aufgerufen wird; er gilt nicht für Parameter, die explizit an die Help routine übergeben werden. Das bedeutet, daß der OPF-Parameter wirkungslos bleibt, falls Sie das Feld, für das Hilfe aufgerufen wird, auch noch explizit als an die Help routine zu übergebenden Parameter angegeben haben.

PC — Periodic Group Count (Anzahl der Periodengruppen-Ausprägungen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
PC	1 bis 99	1	DISPLAY FORMAT INPUT PRINT WRITE

Dieser Parameter darf nur im Reporting Mode verwendet werden.

Mit diesem Session-Parameter geben Sie an, wieviele Ausprägungen einer Periodengruppe oder eines Feldes, das Teil einer Periodengruppe ist, ausgegeben werden sollen, wenn die Periodengruppe bzw. das Feld ohne Index in einem DISPLAY- oder WRITE-Statement angegeben ist.

Beispiel:

```
FORMAT PC=5
```

PD — NATPAGE Page Dataset

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
PD	0 bis 255	50	SET GLOBALS

Dieser Parameter steht nur auf Großrechnern und UNIX/OpenVMS zur Verfügung.

Mit diesem Session-Parameter bestimmen Sie die maximale Anzahl der Seiten (Schirme), die mit der Utility NATPAGE aufgezeichnet werden können.

Wird diese Anzahl überschritten, wird durch jeden weiteren Schirm ein bereits aufgezeichneter überschrieben, wobei die ältesten, d.h. zuerst aufgezeichneten, Schirme nach und nach überschrieben werden (“Wrap-Around”-Verfahren).

Die aufgezeichneten Schirme werden in der Natural-Systemdatei FUSER gespeichert.

Weitere Informationen zur NATPAGE-Utility finden Sie unter den Terminalkommandos %E, %I, %O, %P und %S.

Beispiel:

```
SET GLOBALS PD=60
```

PM — Print Mode (Druck-Modus)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
PM	C, D, I, N	keiner	DEFINE DATA DISPLAY FORMAT INPUT PRINT SET GLOBALS* WRITE

* Mit dem SET GLOBALS-Statement wird der Natural-**Profilparameter** PM (der in Ihrer *Natural Referenz-Dokumentation* beschrieben ist) gesetzt, nicht der **Session-Parameter** PM.

Mit diesem Session-Parameter bestimmen Sie, wie Felder angezeigt werden sollen.

PM=C	(nur möglich auf Großrechnern) Es wird ein alternativer Zeichensatz verwendet (siehe Modul NATPM in der Natural-Source-Library).
PM=D	(nur möglich auf Großrechnern) Definiert DBCS-Felder ohne Shift-Codes (siehe Support of Double-Byte Character Sets in der <i>Natural Operations Documentation for Mainframes</i>).
PM=I	Feldwerte werden in invertierter Richtung, d.h. von rechts nach links, angezeigt (z.B. für Länder des Nahen Ostens).
PM=N	Von der Anzeige kann keine "Hardcopy" gemacht werden.

Beispiel:

```
LIMIT 1
READ EMPLOYEES
DISPLAY NOTITLE NAME
DISPLAY NOTITLE NAME (PM=I)
DISPLAY NOTITLE NAME
END
```

NAME	
MORENO	
	ONEROM
MORENO	

*Anmerkung:**Es ist möglich, mehrere Werte anzugeben.*

PS — Page Size (Seitenlänge)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
PS	10 bis 250, 0 (siehe unten)	physische Seitenlänge	DISPLAY FORMAT INPUT SET GLOBALS WRITE

Mit diesem Session-Parameter bestimmen Sie, wieviele Zeilen eine von einem DISPLAY- oder WRITE-Statement erzeugte Reportseite höchstens haben darf.

Wird PS=0 für den ersten auszugebenden Report (Report 0) gesetzt, so wird die physische Seitenlänge des verwendeten Geräts abzüglich 1 Zeile genommen.

Wird PS=0 für die Reports 1 – 31 gesetzt, so wird damit die automatische Newpage-Verarbeitung unterdrückt, d.h. es werden keine automatischen Seitenvorschub-Verarbeitungen ausgeführt.

PS=0 kann nur mit einem FORMAT-Statement gesetzt werden.

Beispiel:

```
FORMAT PS=40
```

REINP — Interner REINPUT bei ungültigen Daten

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
REINP	ON / OFF	ON	SET GLOBALS

Standardmäßig führt Natural automatisch ein internes REINPUT-Statement aus, wenn auf ein INPUT-Statement hin ungültige Daten eingegeben werden. Mit diesem Session-Parameter können Sie diesen Automatismus ausschalten. Dadurch haben Sie die Möglichkeit, solche Eingabefehler in Ihrer Anwendung selbst zu verarbeiten.

REINP=ON	Bei Eingabe ungültiger Daten wird ein internes REINPUT-Statement ausgeführt.
REINP=OFF	Bei Eingabe ungültiger Daten wird kein internes REINPUT-Statement ausgeführt.

SA — Sound Terminal Alarm (Terminal-Warnton)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
SA	ON / OFF	OFF	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, ob die Warnton-Funktion des Terminals eingesetzt werden soll.

Falls die verwendete Terminal-Hardware die Möglichkeit eines Warntones nicht vorsieht, wird der Parameter ignoriert.

SA=ON	Der Warnton wird jedesmal ausgelöst, wenn der Benutzer von Natural zu einer Eingabe aufgefordert wird.
SA=OFF	Der Warnton wird zur Eingabeaufforderung nicht ausgelöst. Allerdings kann der Warnton nach wie vor über die SOUND ALARM-Klausel eines REINPUT-Statements ausgelöst werden.

Beispiel:

```
SET GLOBALS SA=ON
```

SF — Spacing Factor (Spaltenabstand)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
SF	1 bis 30	1	DISPLAY FORMAT SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, wieviele Leerstellen zwischen zwei Ausgabespalten eines mit einem DISPLAY-Statement erzeugten Natural-Reports standardmäßig eingefügt werden sollen.

Es ist nicht möglich, SF auf "0" zu setzen, da zwischen zwei Report-Spalten immer mindestens eine Spalte frei bleiben muß.

Beispiel:

```
GLOBALS SF=5
```

SG — Sign Position (Vorzeichen-Stelle)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
SG	ON / OFF	ON	DISPLAY FORMAT INPUT PRINT WRITE

Mit diesem Session-Parameter bestimmen Sie, ob einem numerischen Feld eine zusätzliche Stelle zur Anzeige des Vorzeichens vorangestellt werden soll.

Wenn SG=OFF gesetzt ist, werden negative Werte ohne das Vorzeichen “-” ausgegeben.

Beispiel:

```
FORMAT SG=OFF
```

Ist der EM-Parameter angegeben, so überschreibt er den SG-Parameter.

SL — Sourcecode-Zeilenlänge

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
SL	20 bis 80	72	GLOBALS-Systemkommando

Dieser Parameter gilt nur für Natural auf Großrechnern.

Die mit diesem Session-Parameter angegebene Zahl bestimmt, wieviel Stellen eine Natural-Sourcecode-Zeile maximal lang sein darf; dies gilt nur für den EDT-Modus (der mit dem Systemkommando EDT aktiviert wird).

Beispiel:

```
GLOBALS SL=74
```

SM — Structured Mode

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
SM	ON / OFF	OFF	GLOBALS-Systemkommando

Mit diesem Session-Parameter bestimmen Sie, ob im *Structured Mode* programmiert werden muß oder nicht.

Ist der entsprechende Profilparameter vom Natural-Administrator bei der Natural-Installation auf SM=ON gesetzt worden, kann der Session-Parameter SM=OFF nicht angegeben werden.

Andernfalls kann der Zwang zum strukturierten Programmieren mit dem Session-Parameter SM an- und abgeschaltet werden.

Beispiel:

```
GLOBALS SM=ON
```

Anmerkung:

Falls Natural Security installiert ist, kann es sein, daß dieser Parameter im Security-Profil einer Library deaktiviert ist; ist dies der Fall, so gilt für die Library unabänderlich Structured Mode.

SYMGEN — Generate Symbol Table Symboltabelle generieren

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
SYMGEN	ON / OFF	OFF	GLOBALS-Systemkommando

Dieser Session-Parameter steht nur auf UNIX/OpenVMS und Windows zur Verfügung.

Mit diesem Session-Parameter bestimmen Sie, ob eine Symboltabelle generiert wird (siehe auch Profilparameter SYMGEN).

Die Symboltabelle enthält alle in einem Natural-Programm verwendeten Symbole (z.B. Variablennamen). Sie ist Bestandteil des generierten Programms und ist für den Natural Debugger und den Dialog-Editor erforderlich.

Beispiel:

```
GLOBALS SYMGEN=ON
```

TC — Trailing Characters (Nachgestellte Zeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
TC	jedes Zeichen	keiner	DISPLAY FORMAT

Die mit diesem Session-Parameter angegebene Zeichenkette wird bei einem Feld, das über ein DISPLAY-Statement ausgegeben wird, unmittelbar hinter dem Feld angezeigt. Die Breite der Ausgabespalte vergrößert sich dadurch entsprechend.

Sie können eine Zeichenkette von 1 bis 10 Zeichen definieren. Sie können die Zeichenkette wahlweise in Apostrophen angeben; in diesem Fall darf die Zeichenkette jedes beliebige Zeichen enthalten.

Eine Zeichenkette, die Anführungszeichen oder ein schließendes Klammerzeichen enthält, muß in Apostrophen stehen.

Beispiele:

```
FORMAT TC=*
```

```
DISPLAY (TC='*B*')
```

TS — Konvertierung von Systemdatei-Programmausgaben

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
TS	ON / OFF	OFF	GLOBALS-Systemkommando

Dieser Parameter gilt nur für Natural auf Großrechnern.

Dieser Session-Parameter dient dazu, Ausgaben von Natural-Systemdateien (d.h. Dateien, deren Namen mit "SYS" anfangen) mittels einer Umsetzungstabelle umzusetzen. Dies ist gegebenenfalls bei nicht-standardmäßiger Verwendung von Kleinbuchstaben (z.B. bei Ländern des Nahen Ostens) erforderlich; vgl. Session-Parameter PM (Seite 172).

Anmerkung:

Bei TS=ON werden der Profilparameter LC=OFF sowie der Session-Parameter AD=T, die Eingaben in Großbuchstaben umsetzen, ignoriert, da sie eine unerwünschte Zeichenumsetzung für spezielle Zeichensätze bewirken würden.

UC — Underlining Character (Unterstreichungszeichen)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
UC	jedes Zeichen, OFF	– (Bindestrich)	DISPLAY FORMAT

Das mit diesem Session-Parameter angegebene Zeichen wird als Unterstreichungszeichen verwendet für:

- Spaltenüberschriften, die von DISPLAY-Statements generiert werden
- Seitenüberschriften/-unterschriften, die über WRITE TITLE/TRAILER-Statements mit UNDERLINED-Option erzeugt werden.

Falls Sie keine Unterstreichung von Spaltenüberschriften wünschen, haben Sie zwei Möglichkeiten:

- “**UC=** ” — Statt einer Unterstreichung wird eine Leerzeile ausgegeben.
- **UC=OFF** — Die Feldwerte werden unmittelbar unter der Überschrift, ohne Leerzeile dazwischen, ausgegeben.
Sie können UC=OFF nur auf Statement-Ebene eines DISPLAY-Statements angeben; in diesem Fall können Sie für einzelne Felder in dem Statement keine anderen UC-Angaben machen.

Beispiele:

```
FORMAT UC=*
```

```
DISPLAY (UC= ) NAME AGE (UC=+)
```

WH — Warten auf Datensatz im “Hold”

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
WH	ON / OFF	OFF	SET GLOBALS

Dieser Parameter gilt nur für Adabas-Datenbanken.

Mit diesem Session-Parameter bestimmen Sie, was geschehen soll, wenn ein angeforderter Datensatz von einem anderen Benutzer ins “Hold” gestellt wurde.

WH=OFF	Eine Fehlermeldung wird ausgegeben, falls auf einen der gewünschten Datensätze nicht zugegriffen werden kann.
WH=ON	Der Benutzer wird solange in den Wartestatus versetzt, bis entweder der angeforderte Datensatz wieder zur Verfügung steht oder eine Datenbanksystem-Zeitüberschreitung oder Überschreitung eines anderen Limits zur Ausgabe einer entsprechenden Fehlermeldung führt.

Weitere Informationen zur “Hold”-Logik finden Sie im *Natural Leitfaden zur Programmierung*.

Beispiel:

```
SET GLOBALS WH=ON
```

ZD — Zero Division (Teilung durch Null)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
ZD	ON / OFF	ON	SET GLOBALS

Mit diesem Session-Parameter bestimmen Sie, was im Falle einer Division durch Null geschehen soll.

ZD=ON	Natural gibt eine Fehlermeldung aus, falls versucht wird, eine Zahl durch "0" zu teilen.
ZD=OFF	Natural gibt bei einer Division durch "0" als Ergebnis eine "0" aus.

Beispiel:

```
SET GLOBALS ZD=OFF
```

ZP — Zero Printing (Anzeige von Nullwerten)

Parameter	Mögliche Werte	Standardwert	Mögliche Statements
ZP	ON / OFF	ON	DISPLAY FORMAT INPUT PRINT REINPUT SET GLOBALS WRITE

Mit diesem Session-Parameter können Sie die Anzeige eines numerischen Feldes (Format N, P, I oder F) oder eines Zeitfeldes (Format T), dessen Wert aus lauter Nullen besteht, unterdrücken.

ZP=ON	Bei einem Feldwert, der aus lauter Nullen besteht, wird bei numerischen Feldern <i>eine</i> Null bzw. bei Zeitfeldern der ganze Feldwert angezeigt.
ZP=OFF	Ein Feldwert, der aus lauter Nullen besteht, wird <i>nicht</i> angezeigt.

Beispiel:

```
FORMAT ZP=OFF
```

SYSTEMVARIABLEN

Dieses Kapitel beschreibt die Natural-Systemvariablen. Systemvariablen dienen dazu, Systeminformationen auszugeben. Sie können an beliebiger Stelle in einem Programm referenziert werden.

- Alphabetische Liste der Variablen
- Datums- und Zeit-Systemvariablen.

Inhalt modifizierbar: Dies zeigt an, ob Sie einer Systemvariablen in einem Natural-Programm einen anderen Wert zuweisen, d.h. ihren von Natural generierten Inhalt überschreiben können.

Weitere Informationen entnehmen Sie dem Kapitel **Große und dynamische Variablen/Felder** in diesem Handbuch.

Alphabetische Liste der Variablen

Die folgenden Systemvariablen stehen zur Verfügung:

*APPLIC-ID	*ERROR-NR	*LIBRARY-ID	*PARAM-USER	*USER
*APPLIC-NAME	*ERROR-TA	*LINE-COUNT	*PATCH-LEVEL	*USER-NAME
*COM	*ETID	*LINESIZE	*PF-KEY	*WINDOW-LS
*CONTROL	*EVENT	*LOG-LS	*PF-NAME	*WINDOW-POS
*CONVID	*GROUP	*LOG-PS	*PID	*WINDOW-PS
*CPU-TIME	*HARDCOPY	*MACHINE-CLASS	*PROGRAM	*WINMGR
*COUNTER	*HARDWARE	*NATVERS	*ROWCOUNT	*WINMGRVERS
*CURS-COL	*HOSTNAME	*NET-USER	*SCREEN-IO	
*CURS-FIELD	*INIT-ID	*NUMBER	*SERVER-TYPE	
*CURS-LINE	*INIT-PROGRAM	*OCCURRENCE	*STARTUP	
*CURSOR	*INIT-USER	*OPSYS	*STEPLIB	
*DATA	*ISN	*OS	*SUBROUTINE	
*DEVICE	*LANGUAGE	*OSVERS	*THIS-OBJECT	
*DIALOG-ID	*LENGTH	*PAGESIZE	*TPSYS	
*ERROR-LINE	*LEVEL	*PAGE-NUMBER	*UI	

*APPLIC-ID

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die ID der Library (Bibliothek), in der sich der Benutzer gerade befindet.

*APPLIC-NAME

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Wenn Natural Security installiert ist, enthält diese Variable den Namen der Library (Bibliothek), in der sich der Benutzer gerade befindet.

Wenn Natural Security nicht installiert ist, enthält diese Variable den Namen "SYSTEM".

*COM

Format/Länge: A128.

Inhalt modifizierbar: Ja.

Diese Systemvariable stellt einen Kommunikationsbereich dar, der es ermöglicht, Daten von außerhalb eines Bildschirmfensters zu verarbeiten.

Wenn ein Fenster aktiv ist, können normalerweise keine Daten außerhalb des Fensters auf dem Bildschirm eingegeben werden. Wenn jedoch eine Map *COM als modifizierbares Feld enthält, kann ein Benutzer in dieses Feld auch dann Daten eingeben, wenn gerade ein Fenster auf dem Bildschirm aktiv ist.

Die weitere Verarbeitung kann dann vom Inhalt von *COM abhängig gemacht werden. Auf diese Weise können Sie Benutzeroberflächen implementieren, bei denen ein Benutzer immer Daten in der Kommandozeile eingeben kann, selbst wenn ein Fenster mit eigenen Eingabefeldern aktiv ist.

Anmerkung:

*Obwohl *COM als modifizierbares Feld in einem INPUT-Statement verwendet werden kann, wird es **nicht** als Eingabefeld, sondern als Systemvariable behandelt; d.h. Eingaben, die in *COM gemacht werden, werden genommen, wie sie sind, ohne daß eine Eingabeverarbeitung (z.B. Umsetzung in Großbuchstaben) erfolgt.*

*Sobald *COM über ein INPUT-Statement auf dem Bildschirm angezeigt wurde, wird mit jedem anschließenden INPUT- oder REINPUT-Statement der jeweils aktuelle Inhalt von *COM ausgegeben.*

*CONTROL

Format/Länge: Handle.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht nur unter Windows 98 und Windows NT/2000 zur Verfügung.

Diese Systemvariable enthält die Handle des Dialog-Elements, für das der aktuelle Event ausgelöst wurde.

Näheres zu Events finden Sie unter **Event-Driven Programming Techniques** im *Natural User's Guide for Windows*.

*CONVID

Format/Länge: I4.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Konversations-ID des aktuellen konversationalen Remote Procedure Calls (RPC). Diese ID wird von einem OPEN CONVERSATION-Statement gesetzt.

Mit einem OPEN CONVERSATION-Statement kann ein Client einen Server zur alleinigen Benutzung bekommen, um eine Reihe von Services (Subprogrammen) innerhalb eines Server-Prozesses auszuführen. Diese alleinige Benutzung heißt Konversation.

Das OPEN CONVERSATION-Statement dient dazu, eine Konversation zu eröffnen und die an ihr beteiligten Subprogramme anzugeben. Wenn ein OPEN CONVERSATION-Statement ausgeführt wird, weist es der Systemvariablen *CONVID eine eindeutige ID zu, die die Konversation identifiziert.

Es können mehrere Konversationen gleichzeitig offen sein. Um von einer offenen Konversation zu einer anderen zu wechseln, weisen Sie *CONVID die entsprechende Konversations-ID zu.

Weitere Informationen zu Natural RPC finden Sie in Ihrer *Natural RPC Documentation*.

*COUNTER

Format/Länge: P10.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Anzahl, wie oft eine mit einem FIND-, READ- oder HISTOGRAM-Statement initiierte Verarbeitungsschleife durchlaufen wurde.

Durch Angabe von (*r*) in Klammern hinter *COUNTER können Sie eine bestimmte Schleife referenzieren, wobei *r* das Statement-Label bzw. die Sourcecode-Zeilenummer des betreffenden FIND-, READ- oder HISTOGRAM-Statements ist. Wenn Sie keine bestimmte Schleife referenzieren, bezieht sich *COUNTER auf die gerade aktive Verarbeitungsschleife.

Datensätze, die aufgrund einer WHERE-Klausel nicht weiterverarbeitet werden, werden von *COUNTER nicht mitgezählt. Datensätze, die aufgrund eines ACCEPT/REJECT-Statements nicht weiterverarbeitet werden, werden mitgezählt.

*CPU-TIME

Format/Länge: I4.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

*CPU-TIME enthält die aktuell vom Natural-Prozeß benutzte CPU-Zeit.

Warnung:

Aufgrund von CPU-Zeituhrbeschränkungen auf einigen UNIX-Systemen kann der Wert nach nur 4295 Sekunden der CPU-Zeit (etwa 72 Minuten) wieder zurückgesetzt werden.

*CURS–COL

Format/Länge: P3.

Inhalt modifizierbar: Ja (aber es darf kein negativer Wert zugewiesen werden).

Diese Systemvariable enthält die Nummer der Spalte, in der sich der Cursor zur Zeit befindet.

Die Cursor-Position bezieht sich auf das aktive Natural-Bildschirmfenster, unabhängig von seiner physischen Platzierung auf dem Bildschirm. Die Position des Cursors wird ausgehend von Zeile 1 / Spalte 1 der *logischen* Seite bestimmt.

Wenn *CURS–COL einen negativen Wert enthält, bedeutet dies, daß der Cursor außerhalb des aktiven Fensters steht. Wenn *CURS–COL negativ ist, enthält auch *CURS–LINE einen negativen Wert. In diesem Fall bestimmen die *absoluten* Werte beider Systemvariablen die Position des Cursors auf dem *physischen* Bildschirm.

Anmerkung:

Meldungszeile, Funktionstastenleiste und Statistikzeile/Infoline werden nicht als Zeilen mitgezählt.

*CURS–FIELD

Format/Länge: I4.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die interne Identifikation des Feldes, in dem sich der Cursor zur Zeit befindet.

*CURS–FIELD kann nicht alleine, sondern nur in Verbindung mit der POS-Funktion verwendet werden. Sie können beide zusammen dazu benutzen, zu prüfen, ob sich der Cursor gerade in einem bestimmten Feld befindet, und die weitere Verarbeitung von dieser Bedingung abhängig machen. Nähere Informationen siehe POS-Funktion.

Wenn der Cursor nicht in einem Feld steht oder kein REINPUT möglich ist, enthält *CURS–FIELD den Wert "0".

Anmerkung:

*Der Wert von *CURS–FIELD dient nur zur internen Identifikation des Feldes und kann nicht für arithmetische Operationen verwendet werden.*

*CURS–LINE

Format/Länge: P3.

Inhalt modifizierbar: Ja (aber es darf kein negativer Wert oder 0 zugewiesen werden).

Diese Systemvariable enthält die Nummer der Zeile, in der sich der Cursor zur Zeit befindet. Die Cursor-Position bezieht sich auf das aktive Natural-Bildschirmfenster, unabhängig von seiner physischen Plazierung auf dem Bildschirm. Die Position des Cursors wird ausgehend von Zeile 1 / Spalte 1 der logischen Seite bestimmt.

Anmerkung:

Meldungszeile, Funktionstastenleiste und Statistikzeile/Infoline werden nicht als Zeilen mitgezählt.

*CURS–LINE kann auch einen der folgenden Werte enthalten:

Wert	Cursor-Position
0	Auf der oberen oder unteren horizontalen Rahmenzeile eines Bildschirmfensters.
-1	Auf der Natural-Meldungszeile.
-2	Auf der Natural-Statistikzeile/Infoline.
-3	Auf der oberen (Tastennummern)-Funktionstastenzeile.
-4	Auf der unteren (Tastennamen)-Funktionstastenzeile.

Wenn *CURS–COL einen negativen Wert enthält — was bedeutet, daß der Cursor außerhalb des aktiven Fensters steht —, dann enthält auch *CURS–LINE einen negativen Wert. In diesem Fall bestimmen die *absoluten* Werte beider Systemvariablen die Position des Cursors auf dem *physischen* Bildschirm.

*CURSOR

Format/Länge: N6.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Position des Cursors auf dem Eingabebildschirm, wenn die EINGABE-Taste oder eine Funktionstaste gedrückt wird.

Anmerkung:

*Es empfiehlt sich, anstelle von *CURSOR die Systemvariablen *CURS-LINE und *CURS-COL zu verwenden. *CURSOR ist nur noch aus Gründen der Kompatibilität zu früheren Natural-Versionen verfügbar.*

*DATA

Format/Länge: N3.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Anzahl der im Natural-Stack gelagerten Datenelemente, die dem nächsten INPUT-Statement als Eingabedaten zur Verfügung stehen. Ist der Stack leer, enthält *DATA den Wert "0". Ein Wert von "-1" bedeutet, daß das zuoberst im Stack gelagerte Element ein Kommando oder der Name einer Natural-Transaktion ist.

Die Werte der Parameter IA und ID zum Zeitpunkt der Ausführung des STACK-Statements dienen dazu, den Wert von *DATA zu bestimmen.

*DEVICE

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Typ des Geräts, von dem Natural aufgerufen wurde. Sie kann einen der folgenden Werte enthalten:

BATCH	Batch-Betrieb.
COLOR	3279-Kompatibilität. 3278-Bildschirm (Gerät mit erweiterter Attribut-Unterstützung).
VIDEO	3270-Bildschirm, PC-Bildschirm, DEC-VT-Terminal oder ein beliebiger UNIX-Terminaltyp.
TTY	Teletyp oder anderer Start/Stop-Typ.
PC	Benutzung von Natural Connection ist aktiviert (durch Profilparameter PC=ON bzw. Terminalkommando %+).
BTX	BTX-Gerät.
SPOOL	3270-Drucker.
ASYNCH	Asynchrone Session.

*DIALOG-ID

Format/Länge: I4.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht nur unter Windows 98 und Windows NT/2000 zur Verfügung.

Diese Systemvariable enthält die Kennung (ID) der aktuellen Instanz des Dialogs.

Näheres zu Dialogen finden Sie unter **Event-Driven Programming Techniques** im *Natural User's Guide for Windows*.

*ERROR–LINE

Format/Länge: N4.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Sourcecode-Zeilenummer des Statements, das einen Fehler verursacht hat.

*ERROR–NR

Format/Länge: N7.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Fehlernummer des Fehlers, der die Verzweigung zu einer ON ERROR-Bedingung bewirkt hat.

Normalerweise enthält *ERROR–NR die Natural-System-Fehlernummer, die das Eintreten der Fehlerbedingung verursacht hat; wenn allerdings ein “REINPUT WITH TEXT **nnnn*”-Statement ausgeführt wird, wird *ERROR–NR mit der betreffenden *anwendungsspezifischen* Fehlernummer *nnnn* gefüllt.

Sie können dieser Systemvariablen in einem Natural-Programm einen anderen Wert zuweisen, allerdings nicht innerhalb eines ON ERROR-Statement-Blocks.

*ERROR–NR wird wieder auf “0” gesetzt, sobald ein neues Level-1-Programm ausgeführt wird.

*ERROR-TA

Format/Länge: A8.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält den Namen des Programms, an das die Kontrolle im Falle einer Fehlerbedingung übergeben wird.

Im Falle eines Fehlers führt Natural ein STACK TOP DATA-Statement aus und legt folgende Informationen, welche von einer Error-Transaktion als INPUT-Daten verwendet werden können, oben auf dem Stack ab: Fehlernummer (N4 bei SG=OFF, N5 bei SG=ON), Zeilennummer (N4), Status (A1), Programmname (A8), Level (N2). Bei Status "C" oder "L" ist die Zeilennummer "0". Status kann sein:

C	Kommandoverarbeitungsfehler
L	Logon-Fehler.
O	Objektzeit-Fehler
S	Nicht korrigierbarer Syntaxfehler.
R	Fehler auf Remote-Server (im Zusammenhang mit Natural RPC).

*ETID

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Wert zur Identifizierung von Transaktionsdaten (End of Transaction ID) für Adabas. Dieser Wert kann folgendes sein:

- der Wert des Natural-Profilparameters ETID
- die vom TP-Monitor übergebene User-ID (Benutzerkennung) (nur auf Großrechnern)
- der bei der Natural-Initialisierung im User Exit angegebene Wert (nur auf Großrechnern)
- die im Security-Profil des gerade aktiven Benutzers definierte ETID (falls Natural Security eingesetzt wird).

*EVENT

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht nur unter Windows 98 und Windows NT/2000 zur Verfügung.

Diese Systemvariable enthält den Namen des aktuellen Events.

Näheres zu Events finden Sie unter **Event-Driven Programming Techniques** im *Natural User's Guide for Windows*.

*GROUP

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable ist nur unter Natural Security relevant. Sie enthält die ID, über die der Benutzer an eine geschützte Library (Bibliothek) angemeldet ist, also die ID, über die er/sie an die Library gelinkt ist. Dies ist entweder die ID der Gruppe, über die der Benutzer gelinkt ist, oder die User-ID des Benutzers selbst (falls er/sie direkt an die Library gelinkt ist).

Bei einem Logon in eine ungeschützte Library (wo kein Link verwendet wird) enthält *GROUP keinen Wert.

Wenn Natural Security nicht aktiv ist, enthält *GROUP keinen Wert.

*HARDCOPY

Format/Länge: A8.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Kennung des Druckers, der verwendet wird, wenn das Terminalkommando %H eingegeben wird.

*HARDWARE

Format/Länge: A16.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen der Hardware-Plattform, auf der Natural läuft (z.B.: AT&T). Dieser Wert wird vom Betriebssystem geliefert.

*HOSTNAME

Format/Länge: A64.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

Der Name der Maschine, auf der Natural läuft, die mit der vom OS-Kommando "hostname" zurückgegebenen identisch ist.

*INIT-ID

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Unter OpenVMS und UNIX

*INIT-ID enthält die Kennung (ID) des Geräts, von dem aus Natural aufgerufen wurde.

Unter Windows 98 und WindowsNT/2000

*INIT-ID enthält den Wert "PC_WIN".

Auf Großrechnern

*INIT-ID enthält die Kennung (Terminal-ID) des Terminals, von dem aus Natural aufgerufen wurde (gemäß den Konventionen des eingesetzten TP-Systems).

Im Batch-Betrieb auf Großrechnern

*INIT-ID enthält den Step-Namen des Natural-Jobs.

Für eine asynchrone Natural-Session unter Com-plete oder UTM

*INIT-ID enthält die Terminalkennung der Task, von der die asynchrone Session gestartet wurde.

Für eine asynchrone Session unter CICS

*INIT-ID enthält die CICS-Task-Nummer der asynchronen Task.

*INIT-PROGRAM

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen des Programms (bzw. der Transaktion), das gerade als Natural ausgeführt wird.

Im Batch-Betrieb unter OS/390

*INIT-PROGRAM enthält den Namen des Jobs, unter dem die Natural-Session läuft.

Unter OpenVMS, UNIX und Windows

*INIT-PROGRAM enthält den Wert "Natural".

*INIT-USER

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Benutzerkennung (User-ID) des Benutzers.

Unter OpenVMS, UNIX und Windows

*INIT-USER enthält den Wert des Profilparameters USER in der verwendeten Parameterdatei. Ist für den USER-Parameter kein Wert angegeben, enthält *INIT-USER die beim OpenVMS- bzw. UNIX-Login verwendete Benutzerkennung bzw. unter Windows die Kennung, zu deren Eingabe Sie beim Starten von Natural aufgefordert wurden (Standardwert: "SAGPC").

Unter UTM

*INIT-USER enthält die für die UTM-Anwendung definierte Benutzerkennung; sind für die UTM-Anwendung keine Benutzerkennungen definiert, entspricht der Wert von *INIT-USER dem von *INIT-ID.

Unter TIAM

Der Wert von *INIT-USER wird durch den Parameter USERID in dem Makro NATTIAM bestimmt: bei USERID=USER oder NO (standardmäßige Einstellung), enthält *INIT-USER den mit dem LOGON-Kommando angegebenen BS2000/OSD-Jobnamen. Wurde kein BS2000/OSD-Jobname angegeben, enthält *INIT-USER das gleiche wie bei USERID=SYSTEM (oder YES), nämlich die BS2000/OSD-Benutzerkennung.

Im Batch-Betrieb auf Großrechnern

*INIT-USER enthält den Namen des Jobs, unter dem die Natural-Session läuft.

Im Batch-Betrieb unter OS/390

Der Wert von *INIT-USER wird durch den Parameter USERID in dem Natural-OS/390-Batch-Interface (Makro NTOS) bestimmt: bei USERID=YES wird der Wert vom Security-Zugriffskontrollblock (ACEE) des Security-Pakets (z.B. RACF oder ACF2) genommen. Wenn kein Security-Paket verwendet wird, wird der Wert des USER-Parameters der Jobkarte genommen. Ist kein USER-Parameter angegeben, ist der Wert von *INIT-USER der gleiche wie bei USERID=NO, nämlich der Name des Jobs, unter dem die Natural-Session läuft.

*ISN

Format/Länge: P10.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Adabas-interne Satz-Nummer (ISN) des Datensatzes, der gerade in einer mit einem FIND- oder READ-Statement initiierten Verarbeitungsschleife verarbeitet wird.

Mit (*r*) in Klammern hinter *ISN können Sie eine bestimmte Schleife referenzieren, wobei *r* das Label bzw. die Sourcecode-Zeilenummer des betreffenden FIND- oder READ-Statements ist. Wenn Sie keine bestimmte Schleife referenzieren, bezieht sich *ISN auf die gerade aktive Verarbeitungsschleife.

Bei einer mit HISTOGRAM initiierten Verarbeitungsschleife enthält *ISN die Nummer der Ausprägung, in der der zuletzt gelesene Wert des gerade verarbeiteten Deskriptors enthalten ist (ist der Deskriptor nicht Teil einer Periodengruppe, enthält *ISN den Wert "0").

*ISN für DL/I- und SQL-Datenbanken

Für DL/I- und SQL-Datenbanken kann *ISN nicht verwendet werden.

*ISN für VSAM

Bei VSAM-Datenbanken kann *ISN nur für ESDS und RRDS verwendet werden. Bei ESDS enthält *ISN die relative Byte-Adresse (RBA) und bei RRDS die relative Satznummer (RRN) des Datensatzes, der gerade in einer mit einem FIND- oder READ-Statement initiierten Verarbeitungsschleife verarbeitet wird.

*LANGUAGE

Format/Länge: 11.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält den Sprachindikator (Sprachcode). Dieser Sprachindikator wird bei Editiermasken von Datumsfeldern sowie bei Natural- oder benutzergeschriebenen Fehlermeldungen, die über INPUT- oder REINPUT-Statements ausgegeben werden, verwendet.

Jedem Sprachcode ist ein einbuchstabiger Code zugewiesen; in sprachabhängigen Anwendungen werden alle "&" in Namen von Objekten (z.B. Maps, Dialoge, Helprountinen, Subprogramme) durch diesen einbuchstabigen Code ersetzt.

Es stehen Ihnen 60 verschiedene Sprachcodes zur Verfügung. Die Codes sind auf den folgenden Seiten aufgeführt.

Einzelheiten zum Einsatz von Sprachcodes siehe **Gestaltung von Benutzeroberflächen im Natural Benutzerhandbuch für Großrechner**.

Den einzelnen Sprachcodes sind die folgenden Sprachen zugeordnet (die rechte Spalte zeigt die jeweiligen einbuchstabigen Codes für sprachabhängige Map-Namen usw.):

Code	Sprache	Code in Map-Namen
Einbyte-Sprachen (Schreibrichtung von links nach rechts) mit lateinischen Kleinbuchstaben:		
1	Englisch	1
2	Deutsch	2
3	Französisch	3
4	Spanisch	4
5	Italienisch	5
6	Niederländisch	6
7	Türkisch	7
8	Dänisch	8
9	Norwegisch	9
10	Albanisch	A
11	Portugiesisch	B
12	Chinesisch Lateinschrift (Taiwan)	C

Code	Sprache	Code in Map-Namen
13	Tschechisch	D
14	Slowakisch	E
15	Finnisch	F
16	Ungarisch	G
17	Isländisch	H
18	Koreanisch	I
19	Polnisch	J
20	Rumänisch	K
21	Schwedisch	L
22	Kroatisch	M
23	Katalanisch	N
24	Baskisch	O
25	Afrikaans	P
Einbyte-Sprachen (Schreibrichtung von links nach rechts) ohne lateinische Kleinbuchstaben:		
26	Bulgarisch	Q
27	Griechisch	R
28	Japanisch (Katakana)	S
29	Russisch	T
30	Serbisch	U
Einbyte-Sprachen (beide Schreibrichtungen) ohne lateinische Kleinbuchstaben:		
31	Arabisch	V
32	Farsi (Iran)	W
33	Hebräisch	X
34	Urdu (Pakistan)	Y
35	(für zukünftige Verwendung reserviert)	Z
36	(für zukünftige Verwendung reserviert)	a
37	(für zukünftige Verwendung reserviert)	b
38	(für zukünftige Verwendung reserviert)	c
39	(für zukünftige Verwendung reserviert)	d

Code	Sprache	Code in Map-Namen
40	(für zukünftige Verwendung reserviert)	e
Vom Benutzer zugewiesene Sprachen:		
41	(Sie können diesem Code eine Sprache zuweisen)	f
42	(Sie können diesem Code eine Sprache zuweisen)	g
43	(Sie können diesem Code eine Sprache zuweisen)	h
44	(Sie können diesem Code eine Sprache zuweisen)	i
45	(Sie können diesem Code eine Sprache zuweisen)	j
46	(Sie können diesem Code eine Sprache zuweisen)	k
47	(Sie können diesem Code eine Sprache zuweisen)	l
48	(Sie können diesem Code eine Sprache zuweisen)	m
49	(Sie können diesem Code eine Sprache zuweisen)	n
50	(Sie können diesem Code eine Sprache zuweisen)	o
Mehrbyte-Sprachen:		
51	Hindi	p
52	Malaiisch	q
53	Thai	r
54	(für zukünftige Verwendung reserviert)	s
55	(für zukünftige Verwendung reserviert)	t
56	(für zukünftige Verwendung reserviert)	u
Doppelbyte-Sprachen:		
57	Chinesisch (Volksrepublik China)	v
58	Chinesisch (Taiwan)	w
59	Japanisch (Kanji)	x
60	Koreanisch	y

*LENGTH

Format/Länge: I4.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

Diese Systemvariable gibt die Länge eines/r dynamischen Natural-Feldes oder -Variable in Bytes zurück. *LENGTH kann nur für dynamische Variablen benutzt werden, um die aktuelle Größe zu erhalten.

*LEVEL

Format/Länge: N2.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Stufennummer (Level Number) des Objekts (Programm, Subprogramm, externe Subroutine, Map, Helproutine oder Dialog), das gerade ausgeführt wird. Stufennummer 1 bezeichnet jeweils ein Hauptprogramm.

Interne Subroutinen werden von *LEVEL nicht berücksichtigt.

*LIBRARY-ID

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die (mit dem LOGON-Kommando angegebene) Kennung der Bibliothek (Library-ID), in der der Benutzer gerade arbeitet.

Diese Systemvariable entspricht der Systemvariablen *APPLIC-ID.

*LINE-COUNT

Format/Länge: P5.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Zeilennummer der aktuellen, d.h. zuletzt ausgegebenen Zeile.

Natural benutzt diese Variable, um die Zeilennummer für die nächste Zeile eines Reports zu bestimmen.

Der Wert von *LINE-COUNT erhöht sich jedesmal, wenn ein DISPLAY-, WRITE-, PRINT-, SKIP- oder INPUT-Statement ausgeführt wird. Der Wert erhöht sich mit jeder ausgegebenen Zeile um "1"; *LINE-COUNT enthält also die Nummer der zuletzt ausgegebenen Zeile auf der aktuellen Seite.

Mit jedem EJECT- und NEWPAGE-Statement wird *LINE-COUNT wieder auf "1" gesetzt (außer bei NEWPAGE WITH TITLE, wo sich der Wert von *LINE-COUNT aus der Anzahl der Zeilen der mit WITH TITLE ausgegebenen Seitenüberschrift ergibt).

Die höchstmögliche Zeilennummer ist 250.

Erzeugt ein Programm mehrere Reports, können Sie mit (*rep*) in Klammern hinter *LINE-COUNT den Report bestimmen, dessen aktuelle Zeilennummer Sie wünschen.

*LINESIZE

Format/Länge: N7.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die physische Zeilenlänge des I/O-Geräts, von dem aus Natural aufgerufen wurde (falls das eingesetzte TP-System diese Informationen liefern kann).

*LOG-LS

Format/Länge: N3.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Zeilenlänge der *logischen Seite*, die mit dem Primär-Report ausgegeben wird.

*LOG-LS gilt nur für den Primär-Report, nicht für etwaige weitere Reports.

*LOG-PS

Format/Länge: N3.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Seitenlänge der *logischen Seite*, die mit dem Primär-Report ausgegeben wird.

*LOG-PS gilt nur für den Primär-Report, nicht für etwaige weitere Reports.

*MACHINE-CLASS

Format/Länge: A16.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen der Maschinenklasse, auf der Natural läuft.

Sie kann einen der folgenden Werte enthalten:

MAINFRAME
PC
UNIX
VMS

*NATVERS

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

Diese Systemvariable enthält die Natural-Version ausschließlich des Patch-Levels.

*NET-USER

Format/Länge: A253.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die komplette User-ID einer authentifizierten Client-Anfrage. Die ID besteht aus dem Domain-Namen und der eigentlichen User-ID.

Standardmäßig ist der Wert von *NET-USER identisch mit dem von *USER.

Wenn ein NaturalX-Server eine authentifizierte Anfrage erhält, wird die User-ID dieser Anfrage an den Server übergeben und in diese Systemvariable gestellt. (Hierzu wird die DCOM-Funktion "CoQueryClientBlanket" benutzt.)

Nachdem der NaturalX-Server die Anfrage verarbeitet hat, wird *NET-USER wieder auf den Wert zurückgesetzt, den sie vor der Anfrage enthielt (d.h. den Wert von *USER).

Eine nicht authentifizierte Anfrage hat keinen Einfluß auf *NET-USER.

In einer Nicht-NaturalX-Server-Umgebung ist der Inhalt von *NET-USER immer der gleiche wie der von *USER.

*NUMBER

Format/Länge: P10.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Anzahl der Datensätze, die mit einem FIND-Statement (unter Erfüllung der WITH-Bedingung) gelesen wurden, oder die Anzahl der ISNs, die mit einem HISTOGRAM-Statement ausgewählt wurden.

Mit (*r*) in Klammern hinter *NUMBER können Sie ein bestimmtes FIND- oder HISTOGRAM-Statement referenzieren, wobei *r* das Label bzw. die Sourcecode-Zeilenummer des Statements ist. Wenn Sie kein bestimmtes Statement referenzieren, bezieht sich *NUMBER auf die gerade aktive FIND- bzw. HISTOGRAM-Schleife.

Anmerkung:

*Wenn *NUMBER den Wert "999999999" enthält, bedeutet dies, daß der Zugriff auf die betreffende Adabas-Datei mit der Adabas-Funktion "Security By Value" geschützt ist.*

*NUMBER bei DL/I-Datenbanken

Bei DL/I-Datenbanken enthält *NUMBER *nicht* die Anzahl der gefundenen Segment-Ausprägungen. *NUMBER enthält "0", wenn keine Segment-Ausprägung das Suchkriterium erfüllt, und den Wert 8,388,607=X'7FFFFF', wenn mindestens eine Segment-Ausprägung das Suchkriterium erfüllt.

*NUMBER bei SQL-Datenbanken

Wenn *NUMBER in Verbindung mit einem FIND NUMBER-Statement ohne eine WHERE-Klausel oder einem HISTOGRAM-Statement verwendet wird, enthält sie die Anzahl der gefundenen Zeilen. Wenn *NUMBER anderweitig bei einer SQL-Datenbank-Tabelle verwendet wird, enthält sie *nicht* die Anzahl der gefundenen Zeilen: werden keine Zeilen gefunden, enthält *NUMBER den Wert "0"; jeder andere Wert bedeutet, daß Zeilen gefunden wurden, aber der Wert steht in keinem Zusammenhang mit der tatsächlichen Anzahl der gefundenen Zeilen.

*NUMBER bei VSAM-Datenbanken

Bei VSAM-Datenbanken enthält *NUMBER die Anzahl der gefundenen Datensätze nur bei einem HISTOGRAM-Statement oder bei einem FIND-Statement, in dem der Operator "EQUAL TO" im Suchkriterium verwendet wird. Bei jedem anderen Operator enthält *NUMBER *nicht* die Anzahl der gefundenen Datensätze: Werden keine Datensätze gefunden, enthält *NUMBER den Wert "0"; jeder andere Wert bedeutet, daß Datensätze gefunden wurden, aber der Wert steht in keinem Zusammenhang mit der tatsächlichen Anzahl der gefundenen Datensätze.

*OCCURRENCE

Format/Länge: I4.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die aktuelle Anzahl der Ausprägungen eines variablen Arrays.

In einer Parameter Data Area können Sie mit der Index-Notation "1:V" ein Array mit einer variablen Anzahl von Ausprägungen definieren (vgl. DEFINE DATA-Statement im *Natural Statements-Handbuch*). Die tatsächliche Anzahl der Ausprägungen eines solchen variablen Arrays wird erst zur Laufzeit bestimmt. Mit *OCCURRENCE können Sie die tatsächliche Anzahl der Array-Ausprägungen ermitteln.

Der Wert von *OCCURRENCE wird zur Laufzeit ermittelt, wenn das Subprogramm bzw. die Subroutine, in der die Systemvariable angegeben ist, ausgeführt wird.

Hinter *OCCURRENCE geben Sie (in Klammern) den Namen an, mit dem das betreffende Array in der Parameter Data Area definiert ist:

```
*OCCURRENCE (#ARRAY)
```

Nach dem Array-Namen können Sie die Nummer *n* der Dimension (1, 2, 3 oder *) angeben, deren Anzahl von Ausprägungen Sie ermitteln möchten:

```
*OCCURRENCE (#ARRAY, n)
```

Ein Stern (*) bedeutet, daß die Anzahl der Ausprägungen *aller* Dimensionen ermittelt wird.

Wenn Sie keine Dimension angeben, wird die 1. Dimension genommen.

Beispiel:

```
DEFINE DATA
  PARAMETER
    1 #ARRAY (A5/1:V)
  LOCAL
    1 #I (I4)
    ...
END-DEFINE
...
FOR #I = 1 TO *OCCURRENCE(#ARRAY)
  ...
END-FOR
...
```

Anmerkung:

**OCCURRENCE darf als *OCC abgekürzt werden.*

Weitere Beispiele siehe Programme/Subprogramme OCC1P und OCC1N sowie OCC2P und OCC2N in Library SYSEXRM.

*OPSYS

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen des eingesetzten Betriebssystems.

Sie kann einen der folgenden Werte enthalten:

ATT_OSX	FACOM/XA	RS_6000
AVIION	FUJI M73	SCO
BS2000	HP_HPUX	SINIX_52
BS2/XS	MSDOS	SINIX_54
BULL/BOS	MS_OS/2	SUN_SOLA
CMS	MVS/ESA	SUN_SUNO
CMS/XA	MVS/XA	UNISYS 5
DEC-OSF/	NCR 3000	UNISYS 6
DOS/VS	OS	VSE/ESA
DPS300	OS/400	WNT-AXP
DRS 6000	OVMS/AXP	WNT-X86
FACOM	OVMS/VAX	

Anmerkung:

*Es empfiehlt sich, statt *OPSYS die Systemvariablen *MACHINE-CLASS, *HARDWARE und *OS zu verwenden, da diese eine genauere Unterscheidung der Umgebung, in der Natural läuft, ermöglichen.*

*OS

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen des Betriebssystems, unter dem Natural läuft (z.B.: HP-UX). Dieser Wert wird vom Betriebssystem geliefert.

*OSVERS

Format/Länge: A16.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Versionsnummer des Betriebssystems, unter dem Natural läuft. Dieser Wert wird vom Betriebssystem geliefert.

*PAGESIZE

Format/Länge: N7.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die physische Seitenlänge des I/O-Geräts, von dem aus Natural aufgerufen wurde (falls das eingesetzte TP-System diese Informationen liefern kann).

*PAGE-NUMBER

Format/Länge: P5.

Inhalt modifizierbar: Ja.

Diese Systemvariable enthält die Nummer der gerade ausgegebenen Seite eines Reports.

Erzeugt ein Programm mehrere Reports, können Sie mit (*rep*) in Klammern hinter *PAGE-NUMBER den Report bestimmen, auf den sich *PAGE-NUMBER beziehen soll.

Natural initialisiert diese Variable erst, wenn die Formatierung des Reports erfolgt, d.h. sie wird erst relevant, wenn das erste FORMAT-, DISPLAY- oder WRITE-Statement ausgeführt wird. Diese Variable kann von einem Natural-Programm modifiziert werden.

Natural benutzt diese Variable, um die Seitennummer für die nächste Seite eines Reports zu bestimmen. Der Wert von *PAGE-NUMBER erhöht sich jedesmal um "1", wenn mit einem DISPLAY-, WRITE-, SKIP- oder NEWPAGE-Statement eine neue Seite erzeugt wird. Ein EJECT-Statement hat keinen Einfluß auf *PAGE-NUMBER.

*PARM–USER

Format/Länge: A253.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

Diese Systemvariable enthält den Namen der aktuell benutzten Parameterdatei.

*PATCH–LEVEL

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

Diese Systemvariable enthält die aktuelle Patch-Level-Nummer als String-Wert.

*PF_KEY

Format/Länge: A4.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Identifikation der Funktionstaste, die zuletzt gedrückt wurde.

*PF_KEY kann einen der folgenden Werte enthalten:

PA1 bis PA3	PA-Tasten 1 bis 3
PF1 bis PF48	PF-Tasten 1 bis 48
ENTR	ENTER- bzw. EINGABE-Taste
CLR	CLEAR- bzw. LÖSCH-Taste
PEN	Lichtstift
PGDN	PAGE DOWN- bzw. BILD-AB-Taste (nur unter OpenVMS und UNIX).
PGUP	PAGE UP- bzw. BILD-AUF-Taste (nur unter OpenVMS und UNIX).

*PF_KEY enthält die Identifikation der Taste nur, falls die Taste auf dem jeweiligen Level aktiviert ist; falls nicht, enthält *PF_KEY den Wert "ENTR".

Anmerkung:

*Wenn Sie abfragen, ob der Wert von *PF_KEY innerhalb eines bestimmten Wertebereichs liegt, berücksichtigen Sie bitte, daß der Wert von *PF_KEY alphanumerisch ist.*

*PF-NAME

Format/Länge: A10.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen der Funktionstaste, die zuletzt gedrückt wurde. Dies ist der Name, der der Taste mit der NAMED-Klausel eines SET KEY-Statements zugewiesen wurde.

Damit haben Sie die Möglichkeit, die Verarbeitung von einem bestimmten Funktionsnamen abhängig zu machen, anstatt von einer bestimmten Taste. Wollen Sie beispielsweise erreichen, daß Benutzer Hilfe durch Drücken von wahlweise PF1 oder PF13 anfordern können, weisen Sie beiden Tasten den Namen "HILFE" zu und knüpfen das Aufrufen der Hilfe an die Bedingung *PF-NAME='HILFE': die Hilfe wird dann aufgerufen, ganz gleich ob der Benutzer sie mit PF1 oder PF13 angefordert hat.

*PID

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Diese Systemvariable steht auf Großrechnern nicht zur Verfügung.

Diese Systemvariable enthält die aktuelle Prozeß-ID als String-Wert.

*PROGRAM

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen des Natural-Objekts, das gerade ausgeführt wird.

*ROWCOUNT

Format/Länge: I4.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Anzahl der Zeilen, die mit einem der Natural-SQL-Statements “searched” DELETE, “searched” UPDATE oder INSERT (mit *select-expression*) in einer Datenbank-Tabelle gelöscht, geändert bzw. hinzugefügt wurden. *ROWCOUNT bezieht sich jeweils auf das zuletzt ausgeführte dieser Statements.

*SCREEN-IO

Format/Länge: L.

Inhalt modifizierbar: Nein.

Diese Systemvariable zeigt an, ob ein Bildschirm-I/O möglich ist oder nicht.

Sie kann einen der folgenden Werte enthalten:

- TRUE = Bildschirm-I/O ist möglich
- FALSE = Bildschirm-I/O ist nicht möglich.

Im Falle einer dialog-orientierten Anwendung wird *SCREEN-IO mit “TRUE” initialisiert.

Wenn Natural als “DB2 Stored Procedures Server” (*SERVER-TYPE=DB2-SP) oder als RPC-Server (*SERVER-TYPE=RPC) gestartet wird, wird *SCREEN-IO auf “FALSE” gesetzt.

Wenn Natural als ein DCOM-Server (*SERVER-TYPE=DCOM) gestartet wird, wird *SCREEN-IO auf “FALSE” gesetzt, während der Server eine von COM/DCOM aufgerufene Methode ausführt.

Wenn *SCREEN-IO auf “FALSE” gesetzt ist und ein INPUT-Statement, das eine Benutzerreaktion erfordert, ausgeführt wird, gibt Natural den Fehler NAT0723 aus.

*SERVER-TYPE

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Diese Systemvariable zeigt an, als welcher Server-Typ Natural gestartet wurde.

Sie kann einen der folgenden Werte enthalten:

DB2-SP	Natural wurde als "DB2 Stored Procedures"-Server gestartet.
DCOM	Natural wurde als "NaturalX DCOM-Server" gestartet.
DEVELOP	Natural wurde als "Natural Development Server" gestartet.
RPC	Natural wurde als "Natural RPC Server" gestartet.

Wenn Natural nicht als Server gestartet wird, wird *SERVER-TYPE auf Leerzeichen gesetzt.

Anmerkung:

SERVER-TYPE bezieht sich auf Natural als Ganzes, **nicht auf das gerade ausgeführte Natural-Programm (das innerhalb eines Server-Naturals als Client- oder Server-Programm ausgeführt werden kann).*

*STARTUP

Format/Länge: A8.

Inhalt modifizierbar: Ja.

Das Programm, dessen Name in dieser Systemvariable steht, wird immer dann ausgeführt, wenn Natural normalerweise die Kommandoingabeaufforderung (NEXT-Prompt bzw. "Direct Command"-Fenster/-Zeile) anzeigen würde.

*STARTUP enthält den Namen des Programms, das in Natural Security als Startup-Transaktion im Security-Profil der betreffenden Library eingetragen ist (außer im Batch-Betrieb; vgl. *Natural Security Documentation*).

Falls keine Startup-Transaktion eingetragen ist oder Natural Security nicht verwendet wird, enthält *STARTUP den Wert des Profilparameters STARTUP (außer auf Großrechnern).

Auf Großrechnern hängt, falls keine Startup-Transaktion eingetragen ist oder Natural Security nicht verwendet wird, der Wert von *STARTUP davon ab, wie der Profilparameter MENU gesetzt ist:

- Wenn MENU=OFF gesetzt ist, ist *STARTUP leer.
- Wenn MENU=ON gesetzt ist, enthält *STARTUP den Namen "MAINMENU", d.h. das Natural-Hauptmenü wird aufgerufen.

Über ein Natural-Programm können Sie *STARTUP einen Wert zuweisen, der dann ihren jeweils vorherigen Inhalt überschreibt.

Anmerkung:

Ein im Batch-Betrieb verwendetes Startup-Programm muß ein FETCH- oder STACK COMMAND-Statement enthalten; sonst kann es zu Fehler NAT9969 kommen.

Falls Sie die Kommandoingabeaufforderung durch Eingabe von "%%" (oder einem gleichwertigen Kommando) aufrufen — entweder in einer Nicht-Security-Umgebung oder in einer Security-Umgebung, in der der Kommando-Modus für die aktuelle Library nicht verboten ist — wird dadurch der Startup-Mechanismus deaktiviert. Um ihn wieder zu aktivieren, müssen Sie sich entweder erneut in die Library begeben oder ein Programm ausführen, das *STARTUP wieder einen Wert zuweist.

In einer Natural-Security-Umgebung, in der der Kommando-Modus für die aktuelle Library verboten ist, bewirkt "%%", daß das Programm, dessen Name in *STARTUP steht, aufgerufen wird.

*STEPLIB

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen der Steplib-Bibliothek, die mit der Natural-Library, in der der Benutzer gerade arbeitet, verknüpft ist.

Wenn Natural Security nicht installiert ist, enthält *STEPLIB:

- unter OpenVMS, UNIX und Windows: den mit dem Profilparameter LSTEP in der verwendeten Parameterdatei angegebenen *STEPLIB-Namen
- auf Großrechnern: den mit dem Profilparameter STEPLIB angegebenen Namen.

Wenn Natural Security installiert ist, kann die Steplib im Security-Profil der betreffenden Library eingetragen werden.

Anmerkung:

*Bei der *STEPLIB-Library wird immer davon ausgegangen, daß sie dieselbe Datenbank-ID und Dateinummer hat wie die aktuelle Library des Benutzers. Es wird davon ausgegangen, daß außer der Library SYSTEM die Libraries mit dem Namen SYSxxx sich in FNAT und andere Libraries sich in FUSER befinden.*

*SUBROUTINE

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen der externen Subroutine, die gerade ausgeführt wird.

*THIS-OBJECT

Format/Länge: HANDLE OF OBJECT.

Inhalt modifizierbar: Nein.

Diese Systemvariable ist nur mit NaturalX verfügbar. Sie ist eine Handle auf das gerade aktive Objekt. Das gerade aktive Objekt benutzt *THIS-OBJECT dazu, entweder seine eigenen Methoden auszuführen oder eine Referenz auf sich selbst an ein anderes Objekt zu übergeben.

*THIS-OBJECT enthält nur dann einen tatsächlichen Wert, wenn gerade eine Methode ausgeführt wird. Ansonsten enthält sie "NULL-HANDLE".

Weitere Informationen siehe *NaturalX Documentation*.

*TPSYS

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Namen des verwendeten TP-Monitors.

Sie kann einen der folgenden Werte enthalten:

AIM/DC
CICS
COMPLETE
IMS/DC
OS/400
SERVSTUB (NDV Server)
TIAM
TSO
TSS
UTM
VM/CMS
VTAM

Im Batch-Betrieb ist *TPSYS leer.

Wenn kein TP-Monitor verwendet wird, enthält *TPSYS den Wert "NONE".

*UI

Format/Länge: A16.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält den Typ der verwendeten Benutzungsoberfläche:

CHARACTER	Zeichen-orientierte Benutzungsoberfläche.
GUI	Graphische Benutzungsoberfläche.

*USER

Format/Länge: A8.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die Benutzerkennung (User-ID), mit der der Benutzer Natural über die Natural-Security-Logon-Prozedur aufgerufen hat.

Wenn der Profilparameter AUTO=ON (Automatic Logon) gesetzt ist oder wenn Natural Security nicht aktiv ist, entspricht der Wert von *USER dem von *INIT-USER.

*USER-NAME

Format/Länge: A32.

Inhalt modifizierbar: Nein.

Wenn Natural Security installiert ist, enthält diese Systemvariable den Namen des gerade aktiven Natural-Benutzers.

Wenn Natural Security nicht installiert ist, enthält diese Systemvariable den Wert "SYSTEM".

*WINDOW-LS

Format/Länge: N3.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die logische Zeilenlänge (ohne Rahmen) des aktiven Natural-Windows. Vgl. DEFINE WINDOW-Statement.

*WINDOW-POS

Format/Länge: N6.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die logische Position der oberen linken Ecke des aktiven Natural-Windows. Vgl. DEFINE WINDOW-Statement.

Die logische Position wird über mehrere Zeilen hinweg ab der Position "0" (obere linke Ecke) in Zeichen gezählt.

*WINDOW-PS

Format/Länge: N3.

Inhalt modifizierbar: Nein.

Diese Systemvariable enthält die logische Seitenlänge (ohne Rahmen) des aktiven Natural-Windows. Vgl. DEFINE WINDOW-Statement.

*WINMGR

Format/Länge: A16.

Inhalt modifizierbar: Nein.

Bei Verwendung einer graphischen Benutzeroberfläche enthält diese Systemvariable den Namen des verwendeten Window-Managers (z.B.: MOTIF oder PM).

Bei Verwendung einer zeichen-orientierten Benutzeroberfläche ist *WINMGR leer.

Die Art der Benutzeroberfläche ergibt sich aus dem Wert der Systemvariablen *UI.

*WINMGRVERS

Format/Länge: A16.

Inhalt modifizierbar: Nein.

Bei Verwendung einer graphischen Benutzeroberfläche enthält diese Systemvariable die Versionsnummer des verwendeten Window-Managers.

Bei Verwendung einer zeichen-orientierten Benutzeroberfläche ist *WINMGRVERS leer.

Die Art der Benutzeroberfläche ergibt sich aus dem Wert der Systemvariablen *UI.

Datums- und Zeit-Systemvariablen

Die unten aufgeführten Datums- und Zeit-Systemvariablen können in den Statements COMPUTE, DISPLAY, MOVE, PRINT und WRITE sowie in logischen Bedingungen verwendet werden.

Der von Natural generierte Inhalt der Datums- und Zeit-Systemvariablen kann nicht verändert werden, d.h. Sie können keiner dieser Variablen in einem Natural-Programm einen anderen Wert zuweisen.

Alle Datumsvariablen enthalten das aktuelle Datum. Das Format des Datums ist bei jeder Variablen anders, wie die folgende Tabelle zeigt.

Datumsvariable	Format/Länge	Datumsformat*
*DATD	A8	DD.MM.YY
*DAT4D	A10	DD.MM.YYYY
*DATE	A8	DD/MM/YY
*DAT4E	A10	DD/MM/YYYY
*DATG	A15	DDMonatsnameYYYY (Gregorianisch)
*DATI	A8	YY-MM-DD
*DAT4I	A10	YYYY-MM-DD
*DATJ	A5	YYDDD (Julianisch)
*DAT4J	A7	YYYYDDD (Julianisch)
*DATN	N8	YYYYMMDD
*DATU	A8	MM/DD/YY
*DAT4U	A10	MM/DD/YYYY
*DATV	A11	DD-MON-YYYY (**)
*DATVS	A9	DDMONYYYY (**)
*DATX	D	internes Datumsformat

* D = Day (Tag), M = Monat, Y = Year (Jahr).

** Die Systemvariablen *DATV und *DATVS stehen auf Großrechnern nicht zur Verfügung.

Zeitvariable	Format/Länge	Erklärung
TIMD(<i>r</i>)	N7	Kann nur in Verbindung mit einem vorangegangenen SETTIME-Statement verwendet werden. Enthält die Zeit, die seit der Ausführung des SETTIME-Statements verstrichen ist (im Format HHIISST ()). Bei mehreren SETTIME-Statements können Sie über Statement-Label bzw. Sourcecode-Zeilenummer (<i>r</i>) ein bestimmtes SETTIME-Statement referenzieren.
TIME	A10	Enthält die aktuelle Uhrzeit im Format HH:II:SS.T ().
*TIME-OUT	N5	Enthält die Anzahl der verbleibenden Sekunden, bevor die aktuelle Transaktion wegen Zeitüberschreitung abgebrochen wird (nur verfügbar unter Natural Security). *TIME-OUT kommt nur zum Tragen, wenn ein FIND-, READ- oder GET-Statement einen Datensatz zum Zwecke des Änderns oder Löschens liest. *TIME-OUT ist "0", wenn kein datenverändernder Datenbankzugriff erfolgt. *TIME-OUT wird auf "0" zurückgesetzt, wenn ein END TRANSACTION- oder BACKOUT TRANSACTION-Statement ausgeführt wird.
*TIMESTMP	B8	Hardware-interner Zeitzähler (Store Clock).
TIMN	N7	Enthält die aktuelle Uhrzeit im Format HHMMSST ().
*TIMX	T	Enthält die aktuelle Uhrzeit im internen Zeitformat.

* H = Hour (Stunde), I = Minute, S = Sekunde, T = Tenth of a second (Zehntelsekunde).

Beispiel für Datums- und Zeit-Systemvariablen:

```

* EXAMPLE 'DATIVAR': DATE AND TIME SYSTEM VARIABLES
*****
WRITE NOTITLE
  'DATE IN FORMAT DD.MM.YYYY ' *DAT4D /
  'DATE IN FORMAT DD/MM/YYYY ' *DAT4E /
  'DATE IN GREGORIAN FORM ' *DATG /
  'DATE IN FORMAT YYYY-MM-DD ' *DAT4I /
  'DATE IN FORMAT YYYYDDD ' *DAT4J /
  'DATE IN FORMAT YYYYMMDD ' *DATN /
  'DATE IN FORMAT MM/DD/YYYY ' *DAT4U /
  'DATE IN INTERNAL FORMAT ' *DATX (DF=L) ///
  'TIME IN FORMAT HH:II:SS.T ' *TIME /
  'TIME IN FORMAT HHIISST ' *TIMN /
  'TIME IN INTERNAL FORMAT ' *TIMX /
*
MOVE *DATX TO #DATE(D)
ADD 14 TO #DATE
WRITE 'CURRENT DATE' *DATX (DF=L) 3X
      'CURRENT DATE + 14 DAYS ' #DATE (DF=L)
MOVE *TIMX TO #TIME(T)
ADD 100 TO #TIME
WRITE 'CURRENT TIME' *TIMX 5X 'CURRENT TIME + 10 SECONDS' #TIME
END

```

DATE IN FORMAT DD.MM.YYYY	19.01.1999	
DATE IN FORMAT DD/MM/YYYY	19/01/1999	
DATE IN GREGORIAN FORM	19 January	1999
DATE IN FORMAT YYYY-MM-DD	1999-01-19	
DATE IN FORMAT YYYYDDD	1999019	
DATE IN FORMAT YYYYMMDD	19990119	
DATE IN FORMAT MM/DD/YYYY	01/19/1999	
DATE IN INTERNAL FORMAT	1999-01-19	
TIME IN FORMAT HH:II:SS.T	13:34:04.4	
TIME IN FORMAT HHIISST	1334044	
TIME IN INTERNAL FORMAT	13:34:04	
CURRENT DATE 1999-01-19	CURRENT DATE + 14 DAYS	1999-02-02
CURRENT TIME 13:34:04	CURRENT TIME + 10 SECONDS	13:34:14

GROSSE UND DYNAMISCHE VARIABLEN/FELDER

In diesem Kapitel werden die folgenden Themen behandelt:

- Allgemeines
- Definition dynamischer Variablen
- Systemvariable *LENGTH(Feld)
- Speichergrenzen-Prüfungen
- Statements EXPAND und REDUCE
- Verwendung dynamischer Variablen

Allgemeines

Ab Natural Version 4.1 verfügt der Benutzer über eine erweiterte Funktionalität bei der Verwendung großer Variablen, wodurch die vorhandenen Speichergrößenbeschränkungen beseitigt werden, und eine dynamische Belegung dieser Variablen zur Ausführungszeit ermöglicht wird.

Große Variablen für alphanumerische und binäre Daten basieren auf den wohlbekanntesten Natural-Formaten A und B. Die Beschränkungen von 253 für Format A und 126 für Format B sind nicht mehr gültig. Die neue Speichergrößenbeschränkung ist 1 GB.

Diese großen statischen Variablen und Felder werden in Bezug auf Definition, Redefinition, Speicherwertebelegung, Konvertierungen, Referenzierung in Statements, usw. genauso behandelt wie die traditionellen alphanumerischen und binären Variablen und Felder. Alle Regeln hinsichtlich alphanumerischer und binärer Formate gelten für diese großen Formate.

Insofern als die maximale Größe großer Datenstrukturen (zum Beispiel Bilder, Klänge, Videos) zur Entwicklungszeit der Anwendung vielleicht noch nicht genau bekannt ist, ist in Natural zusätzlich die Definition alphanumerischer und binärer Variablen mit dem Attribut DYNAMIC vorgesehen. Der Wertespeicher von Variablen, die mit diesem Attribut definiert sind, wird zur Ausführungszeit dynamisch erweitert, wenn dies erforderlich ist; zum Beispiel bei einer Zuweisungsoperation:

```
#picture1 := #picture2
```

Dies bedeutet, daß große binäre und alphanumerische Datenstrukturen in Natural verarbeitet werden können, ohne daß die Definition eines Limits zur Entwicklungszeit erforderlich wäre. Die Ausführungszeit-Belegung dynamischer Variablen unterliegt natürlich den aktuellen Hauptspeicherbeschränkungen.

Wenn die Belegung dynamischer Variablen zu einer vom zugrundeliegenden Betriebssystem zurückgegebenen Fehlerbedingung hinsichtlich eines unzureichenden Hauptspeichers führt, kann das Statement ON ERROR verwendet werden, um diese Fehlerbedingung abzufangen; andernfalls gibt Natural eine Fehlermeldung zurück.

Die Natural-Systemvariable *LENGTH kann verwendet werden, um die Anzahl der Bytes des Wertespeichers zu erhalten, die gerade für eine gegebene dynamische Variable verwendet werden. Bei Zuweisungen mit der dynamischen Variable setzt Natural *LENGTH automatisch auf die Länge des Source-Operanden. *LENGTH(Feld) gibt deshalb die aktuell für ein dynamisches Natural-Feld oder eine dynamische Natural-Variable verwendete Speichergröße in Bytes an.

Wenn der der dynamischen Variable zugewiesene Speicherplatz nicht mehr gebraucht wird, kann das Statement REDUCE DYNAMIC VARIABLE verwendet werden, um die der dynamischen Variable zugewiesene Speichergröße auf Null (oder eine andere Speichergröße Ihrer Wahl) zu reduzieren. Wenn die Obergrenze der Hauptspeicherbelegung für eine bestimmte dynamische Variable bekannt ist, kann das EXPAND-Statement verwendet werden, um die für die dynamische Variable benutzte Speichergröße auf diese bestimmte Speichergröße zu setzen.

Wenn eine dynamische Variable initialisiert werden soll, sollte das Statement MOVE ALL UNTIL zu diesem Zweck verwendet werden.

Anmerkung bezüglich Natural RPC:

Große und dynamische Variablen werden zur Zeit von Natural Remote Procedure Call (RPC) nicht unterstützt.

Definition dynamischer Variablen

Da die tatsächliche Speichergröße für große alphanumerische und binäre Datenstrukturen zur Entwicklungszeit der Anwendung vielleicht noch nicht ganz genau bekannt ist, kann die Definition dynamischer Variablen des Formats A oder B eingesetzt werden, um diese Strukturen verwalten zu können.

Die dynamische Belegung und Erweiterung (Reallokation = Neubelegung) großer Variablen ist für die Anwendungsprogrammier-Logik transparent. Die Definition dynamischer Variablen erfolgt ohne Längenangabe. Die Hauptspeicher-Belegung wird zur Ausführungszeit implizit vorgenommen, wenn die dynamische Variable als Ziel-Operand verwendet wird, oder explizit mit einem EXPAND-Statement.

Dynamische Variablen können nur in einem DEFINE DATA-Statement mittels der folgenden Syntax definiert werden:

```
level  variable-name  ( A )  DYNAMIC
level  variable-name  ( B )  DYNAMIC
```

Eine dynamische Variable kann nur als Skalar definiert werden (es ist keine dynamische Array-Definition möglich).

Eine dynamische Variable muß nicht unbedingt in einer REDEFINE-Klausel stehen, und eine Redefinition einer dynamischen Variable oder einer eine dynamische Variable enthaltenden Gruppe ist nicht möglich. Die Klauseln CONST und INIT können nicht für dynamische Variablen eingesetzt werden.

Systemvariable *LENGTH(Feld)

Die Speichergröße des gerade eingesetzten Wertespeichers für eine dynamische Variable kann mit der Systemvariable *LENGTH ermittelt werden. *LENGTH wird bei Zuweisungen automatisch auf die (eingesetzte) Länge des Source-Operanden gesetzt.

Anmerkung:

Aufgrund von Systemleistungserwägungen kann der zugewiesene Speicher größer als der verwendete Speicher sein. Es ist dem Natural-Programmierer nicht möglich, Informationen über die aktuell zugewiesene Speichergröße zu erhalten. Dies ist ein interner Wert.

*LENGTH(Feld) gibt die verwendete Speichergröße eines dynamischen Natural-Feldes oder einer dynamischen Natural-Variable in Bytes zurück. *LENGTH kann nur zur Ermittlung der gerade für dynamische Variablen benutzten Speichergröße verwendet werden.

Speichergrenzen-Prüfungen

Profilparameter DSLM

Aus Kompatibilitätsgründen kann zur Kompilierungszeit für Variablen fester Länge eine Speichergrenzen-Prüfung mit dem DSLM-Parameter erfolgen. Der DSLM-Parameter begrenzt die Speichergröße auf 32 KB pro Variable.

Profilparameter USIZE

Für dynamische Variablen ist zur Kompilierungszeit eine Speichergrenzen-Prüfung nicht möglich, weil für dynamische Variablen keine Länge definiert ist. Die Größe des Benutzerpuffer-Bereichs, d.h. der User Buffer Area (USIZE) zeigt die Größe des Benutzerpuffers im virtuellen Hauptspeicher an. Der Benutzerpuffer enthält alle von Natural dynamisch zugewiesenen Daten. Wenn eine dynamische Variable zur Ausführungszeit zugewiesen und erweitert wird, und das USIZE-Limit überschritten wird, wird eine Fehlermeldung zurückgegeben.

Statements EXPAND und REDUCE

Die Statements EXPAND und REDUCE werden verwendet, um Hauptspeicher für eine dynamische Variable explizit zu belegen und freizugeben.

Syntax

```
EXPAND [ SIZE OF ] DYNAMIC [ VARIABLE ] operand1 TO operand2  
REDUCE [ SIZE OF ] DYNAMIC [ VARIABLE ] operand1 TO operand2
```

wobei *operand1* eine dynamische Variable und *operand2* ein nicht negativer, numerischer Speichergrößen-Wert ist.

Das EXPAND-Statement wird verwendet, um den mit der dynamischen Variable belegten Speicher auf eine gegebene Größe zu erweitern. Die gerade der dynamischen Variable zugewiesene Speichergröße (*LENGTH) wird nicht geändert.

Wenn die gegebene Speichergröße kleiner als der gerade mit der dynamischen Variablen belegte Speicher ist, wird das Statement ignoriert.

Das REDUCE-Statement dient der Verringerung der Größe der Hauptspeicher-Belegung. Der mit der dynamische Variablen belegte, über die vorgegebene Größe hinausgehende Hauptspeicher wird sofort freigegeben (wenn das Statement ausgeführt wird).

Wenn der gerade mit der dynamischen Variable belegte (mit *LENGTH definierte) Speicher die vorgegebene Größe überschreitet, wird *LENGTH dieser dynamischen Variable auf diese Größe gesetzt. Der Inhalt der Variablen wird abgeschnitten, aber nicht geändert. Wenn die vorgegebene Größe den aktuell mit der dynamischen Variable belegten Speicherplatz überschreitet, wird das Statement ignoriert.

Verwendung dynamischer Variablen

- Zuweisungen mit dynamischen Variablen
- Initialisierung dynamischer Variablen
- Zeichenketten-Manipulation mit dynamischen Alpha-Variablen
- Logische Bedingungen (LCC) mit dynamischen Variablen
- Parameter-Übertragung mit dynamischen Variablen
- Arbeitsdatei-Zugriff mit großen und dynamischen Variablen
- DDM-Generierung und -Bearbeitung für Spalten variabler Länge
- Zugriff auf große Datenbank-Objekte
- Performance-Aspekte bei dynamischen Variablen

Im allgemeinen kann eine dynamische alphanumerische Variable immer dann verwendet werden, wenn ein Operand des Formats A oder des Formats B erlaubt ist.

Ausnahme:

Dynamische Variablen sind innerhalb der Statements DISPLAY, WRITE, PRINT, STACK, INPUT, REINPUT und SORT nicht zulässig. (Um dynamische alphanumerische Variablen ANZUZEIGEN, zu SCHREIBEN oder AUSZUDRUCKEN, muß die Variable mit dem MOVE SUBSTRING-Statement in kleinere Portionen aufgeteilt werden.)

Die für dynamische Variablen verwendete (mit *LENGTH definierte) Länge und die ihnen zugewiesene Speichergröße entspricht Null, bis zu dem Zeitpunkt, an dem zuerst die Variable als Zielerand referenziert wird. Aufgrund von Zuweisungen oder anderer Operationen können dynamische Variablen zuerst zugewiesen oder auf die genaue Größe des Source-Operanden erweitert (reallokiert=neu belegt) werden.

Die Größe einer dynamischen Variable kann erweitert werden, wenn sie als ein veränderlicher Operand (Zieloperand) in folgenden Statements verwendet wird:

- Zieloperand in einer Zuweisung (ASSIGN, MOVE)
- *operand2* in COMPRESS
- *operand1* in EXAMINE REPLACE
- *operand4* in SEPARATE
- READ WORK FILE
- Parameter- oder View-Feld in der INTO-Klausel von SELECT
- CALLNAT, PERFORM (außer wenn AD=O, oder BY VALUE in PDA)
- SEND METHOD

Zur Zeit gibt es die folgenden Begrenzungen bei der Verwendung von großen Variablen:

- CALL-Statement-Parameter gröÙe kleiner als 64 KB pro Parameter (keine Begrenzung für das CALL-Statement mit INTERFACE4-Option).

In den folgenden Abschnitten wird die Verwendung dynamischer Variablen in allen Einzelheiten mit Beispielen erörtert.

Zuweisungen mit dynamischen Variablen

Im allgemeinen erfolgt eine Zuweisung in der gerade verwendeten (mit *LENGTH definierten) Länge des Source-Operanden.

Wenn der Zieloperand eine dynamische Variable ist, wird seine aktuell zugewiesene Speichergröße möglicherweise erweitert, um den Source-Operanden ohne Abschneiden zu verschieben.

Beispiel 1

```
#MyDynText1 := operand or
MOVE operand to #MyDynText1
#MyDynText1 wird automatisch erweitert, bis der Source-Operand kopiert
werden kann.
```

MOVE ALL, MOVE ALL UNTIL mit dynamischen Zieloperanden werden wie folgt definiert:

- MOVE ALL verschiebt den Source-Operanden so oft in den Zieloperanden, bis die verwendete (mit *LENGTH definierte) Länge des Zieloperanden erreicht ist. *LENGTH wird nicht geändert. Wenn *LENGTH Null ist, wird das Statement ignoriert.
- MOVE ALL *operand1* TO *operand2* UNTIL *operand3* verschiebt *operand1* so oft in *operand2*, bis die in *operand3* angegebene Länge erreicht ist. Wenn *operand3* größer als *LENGTH(*operand2*) ist, wird *operand2* erweitert und *LENGTH(*operand2*) wird auf *operand3* gesetzt. Ist *operand3* kleiner als *LENGTH(*operand2*), wird die verwendete Länge auf *operand3* reduziert. Ist *operand 3* gleich *LENGTH(*operand2*), entspricht dessen Verhalten dem von MOVE ALL.

Beispiel 2

```
#MyDynText1 := 'ABCDEFGHIJKLMNO'          /* *LENGTH(#MyDynText1) ist 15
MOVE ALL 'AB' TO #MyDynText1              /* Inhalt von #MyDynText1 ist
                                           / 'ABABABABABABABA';
                                           /* *LENGTH ist noch 15
MOVE ALL 'CD' TO #MyDynText1 UNTIL 6      /* Inhalt von #MyDynText1 ist
                                           / 'CDCDCD';
                                           /* *LENGTH ist 6
MOVE ALL 'EF' TO #MyDynText1 UNTIL 10     /* Inhalt von #MyDynText1 ist
                                           / 'EFEFEFEFEF';
                                           /* *LENGTH ist 10
```

MOVE JUSTIFIED wird zur Kompilierungszeit zurückgewiesen, wenn der Zieloperand eine dynamische Variable ist.

MOVE SUBSTR und MOVE TO SUBSTR sind zulässig. MOVE SUBSTR führt zu einem Laufzeitfehler, wenn ein Substring (Zeichenteilfolge) hinter der verwendeten (mit *LENGTH definierten) Länge einer dynamischen Variablen referenziert wird. MOVE TO SUBSTR führt zu einem Laufzeitfehler, wenn eine Substring-Position hinter *LENGTH + 1 referenziert wird, weil dies zu einer nicht definierten Lücke im Inhalt der dynamischen Variable führen würde. Wenn der Zieloperand mit MOVE TO SUBSTR erweitert werden sollte (zum Beispiel, wenn der zweite Operand auf *LENGTH+1 gesetzt wird), dann ist der dritte Operand zwingend.

Beispiel 3

```

/* valid
#op2 := *LENGTH(#MyDynText1)
MOVE SUBSTR (#MyDynText1, #op2) TO operand          /* letztes Zeichen
                                                    /* in Operanden verschieben

#op2 := *LENGTH(#MyDynText1) + 1
MOVE operand TO SUBSTR(#MyDynText1, #op2, #len_operand)
                                                    /* Operand mit
                                                    /* #MyDynText1 verketteten
                                                    /* ungültig

#op2 := *LENGTH(#MyDynText1) +1
MOVE SUBSTR (#MyDynText1, #op2, 10) TO operand      /* führt zu
                                                    /* Laufzeitfehler;
                                                    /* undefinierter Substring

#op2 := *LENGTH(#MyDynText1 +10)
MOVE operand TO SUBSTR(#MyDynText1, #op2, #len_operand) /* führt zu
                                                    /* Laufzeitfehler;
                                                    /* undefinierte Lücke

#op2 := *LENGTH(#MyDynText1) +1
MOVE operand TO SUBSTR(#MyDynText1, #op2)          /* führt zu
                                                    /* Laufzeitfehler;
                                                    /* undefinierte Länge

```

Zuweisungs-Kompatibilität

Beispiel:

```
#MyDynText1 := #MyStaticVar1
#MyStaticVar1 := #MyDynText2
```

Wenn der Source-Operand eine statische Variable ist, wird die für den dynamischen Zieloperanden definierte Länge (*LENGTH(#MyDynText1)) auf die Formatlänge der statischen Variablen gesetzt und der Source-Operand in dieser Länge kopiert, einschließlich nachgestellter Leerzeichen (Format A) oder Nullen (Format B).

Wenn der Zieloperand statisch und der Source-Operand dynamisch ist, wird die dynamische Variable in ihrer aktuell verwendeten Größe kopiert. Wenn diese Größe kleiner als die Formatlänge der statischen Variablen ist, wird der Rest mit Leerstellen oder Nullen aufgefüllt. Anderenfalls wird der Wert abgeschnitten. Wenn die aktuell verwendete Größe der dynamischen Variablen gleich 0 ist, wird der statische Zieloperand mit Leerstellen oder Nullen aufgefüllt.

Initialisierung dynamischer Variablen

Dynamische Variablen können mit Leerstellen (alphanumerisch) oder Nullen (binär) bis auf die aktuell verwendete (mit = *LENGTH definierte) Länge mit dem RESET-Statement initialisiert werden. *LENGTH wird nicht geändert.

Beispiel 1:

```
DEFINE DATA LOCAL
:
#MyDynText1 (A) DYNAMIC
:
END-DEFINE
:
#MyDynText1 := 'short text'
write *LENGTH(#MyDynText1) /* verwendete Länge ist 10
:
RESET #MyDynText1 /* verwendete Länge ist immer noch 10,
/* Wert entspricht 10 Leerstellen
:
```

Um eine dynamische Variable mit einem angegebenen Wert in einer vorgegebenen Größe zu initialisieren, kann das Statement MOVE ALL UNTIL verwendet werden.

Beispiel 2:

```
:  
MOVE ALL 'Y' TO #MyDynText1 UNTIL 15          /* #MyDynText1 enthält 15 'Y's,  
:                                               /* verwendete Länge ist 15
```

Zeichenketten-Manipulation mit dynamischen Alpha-Variablen

Wenn ein veränderlicher Operand eine dynamische Variable ist, wird seine aktuell zugewiesene Speichergröße möglicherweise erweitert, um die Operation ohne Abschneiden oder Ausgabe einer Fehlermeldung auszuführen. Dies gilt für die Verkettung (COMPRESS) und Aufteilung dynamischer alphanumerischer Variablen (SEPARATE).

Beispiel

```
DEFINE DATA LOCAL
  1 #MyDynText1          (A) DYNAMIC
  1 #MyDynText2          (A) DYNAMIC
  ...
COMPRESS ... INTO #MyDynText2
```

#MyDynText2 wird erweitert, um die Source-Operanden zu komprimieren.
Anmerkung: im Falle von nicht dynamischen Variablen kann der Wert abgeschnitten werden.

```
SEPARATE INTO #MyDynText1 #MyDynText2 WITH DELIMITER
```

#MyDynText1 and #MyDynText2 werden möglicherweise erweitert oder verringert, um den Source-Operanden aufzuteilen.

```
EXAMINE #MyDynText1 FOR REPLACE
```

#MyDynText1 wird möglicherweise erweitert oder verringert, um die REPLACE-Operation mit Erfolg auszuführen.

Anmerkung:

Im Falle von nicht dynamischen Variablen kann eine Fehlermeldung zurückgegeben werden.

Logische Bedingungen (LCC) mit dynamischen Variablen

Im allgemeinen erfolgt eine Lese-Operation (wie z.B. LCC) mit einer dynamischen Variablen in ihrer aktuell verwendeten Speichergröße. Dynamische Variablen werden wie statische Variablen verarbeitet, wenn sie "Read-only" (unveränderlich) verwendet werden.

Beispiel 1

```
IF #MyDynText1 = #MyDynText2 OR #MyDynText1 = ""
IF #MyDynText1 < #MyDynText2 OR #MyDynText1 < ""
IF #MyDynText1 > #MyDynText2 OR #MyDynText1 > ""
```

Auch im Falle von nachfolgenden Leerzeichen oder Nullen zeigen dynamische Variablen ein ähnliches Verhalten.

Bei dynamischen Variablen ist der alphanumerische Wert 'AA' gleich 'AA' und der binäre Wert '00003031' ist gleich '3031'. Wenn ein Vergleichsergebnis nur im Falle einer exakten Kopie wahr (TRUE) sein sollte, müssen die für die dynamischen Variablen verwendeten Längen zusätzlich verglichen werden. Wenn eine Variable eine exakte Kopie der anderen ist, sind ihre verwendeten Längen auch gleich.

Beispiel 2

```
#MyDynText1 := 'hello'          /* verwendete Länge ist 5
#MyDynText2 := 'hello '        /* verwendete Länge ist 10
IF #MyDynText1 = #MyDynText2  /* TRUE
:
IF #MyDynText1 = #MyDynText2
  AND *LENGTH(#MyDynText1) = *LENGTH(#MyDynText2)
  /* FALSE
:
```

Zwei dynamische Variablen werden Stelle für Stelle von links nach rechts bis zu ihren verwendeten Mindestlängen miteinander verglichen. Die erste Stelle, an der die Variablen nicht gleich sind, bestimmt, ob die erste oder zweite Variable größer oder kleiner als die andere oder gleich der anderen ist. Die Variablen sind gleich, wenn sie bis zu ihren verwendeten Mindestlängen gleich sind und der Rest der längeren Variable nur Leerstellen (Format A) oder Nullen (Format B) enthält.

Beispiel 3

```
#MyDynText1 := 'hello1'          /* verwendete Länge ist 6
#MyDynText2 := 'hello2'          /* verwendete Länge ist 10
IF #MyDynText1 #MyDynText2      /* TRUE (wahr) : #MyDynText2
:= "hallo" IF #MyDynText1 > #MyDynText2 /* TRUE (wahr)
:
```

Vergleichskompatibilität

Vergleiche zwischen dynamischen und statischen Variablen entsprechen Vergleichen zwischen dynamischen Variablen. Die Formatlänge der statischen Variablen wird als ihre verwendete Länge interpretiert.

Beispiel

```
#MyStatText1 := 'hello'           /* Formatlänge von MyStatText1 ist 20
#MyDynText1  := 'hello'           /* verwendete Länge ist 5
IF #MyStatText1 = #MyDynText1     /* TRUE (wahr)
:
IF #MyStatText1 > #MyDynText1     /* FALSE (falsch)
```

Parameter-Übertragung mit dynamischen Variablen

Dynamische Variablen können als Parameter an ein aufgerufenes Programm-Objekt (CALLNAT oder PERFORM) übergeben werden. “Call-by-reference” (d.h. Aufruf über Referenzierung) ist möglich, weil der Wertespeicher einer dynamischen Variablen zusammenhängend aufgebaut ist. “Call-by-value” (Aufruf über Wert) bewirkt eine Zuweisung mit der Variablen-Definition des Aufrufers als Source-Operanden und der Parameter-Definition als Zieloperanden. “Call-by-value result” (Aufruf über Wert mit Ergebnis) bewirkt zusätzlich eine Verschiebung in die entgegengesetzte Richtung.

Bei “call-by-reference” müssen beide Definitionen DYNAMIC (dynamisch) sein. Wenn nur eine von ihnen DYNAMIC ist, resultiert daraus ein Laufzeitfehler. Im Falle von “call-by-value (result)” sind alle Kombinationen möglich. Die folgende Tabelle veranschaulicht die gültigen Kombinationen:

Call By Reference

	Parameter	
Aufrufer	Statisch	Dynamisch
Statisch	Yes	Yes
Dynamisch	Yes	Yes

Die Formate dynamischer Variablen A oder B müssen miteinander übereinstimmen.

Call By Value (Result)

	Parameter	
Aufrufer	Statisch	Dynamisch
Statisch	Yes	Yes
Dynamisch	Yes	Yes

Anmerkung:

Im Falle von statischen/dynamischen oder dynamischen/statischen Definitionen kann nach den Datenübertragungsregeln der entsprechenden Zuweisungen ein Abschneiden der Werte erfolgen.

Beispiel 1

```

DEFINE DATA LOCAL
  1 #MyText (A) DYNAMIC
  :
  #MyText := '123456'           /* erweitert auf 6 Bytes
  WRITE *LENGTH(#MyText)      /* ist 6
  CALLNAT 'SUB1' USING #MyText
  WRITE *LENGTH(#MyText)      /* ist 8; zugewiesene Speichergröße ist 8

Subpgm SUB1:
DEFINE DATA PARAMETER
  1 #MyParm (A) DYNAMIC BY VALUE RESULT
  :
  WRITE *LENGTH(#MyParm)      /* ist 6; Wertezwischenspeicher von 6 Bytes wird
                               /* für #MyParm zugewiesen
  #MyParm := '1234567'        /* erweitert auf 7
  #MyParm := '12345678'       /* zugewiesene Speichergröße=8 Bytes
  EXPAND DYNAMIC VARIABLE #MyParam TO 10
  WRITE *LENGTH(#MyParm)      /* ist 8; zugewiesene Speichergröße ist 10
  END                          /* Inhalt von #Myparm wird (zurück) nach #MyText
                               /* verschoben
                               /* verwendete Länge ist 8; #MyText wird auf 8
                               /* erweitert

```

Beispiel 2

```
DEFINE DATA LOCAL
  1 #MyText (A) DYNAMIC
  :
  #MyText := '123456'           /* erweitert auf 6 Bytes
  WRITE *LENGTH(#MyText)      /* ist 6
  CALLNAT 'SUB2' USING #MyText
  WRITE *LENGTH(#MyText)      /* ist 8; zugewiesene Speichergröße
                               /* ist 10 (erweitert in SUB2)

Subpgm SUB2:
DEFINE DATA PARAMETER
  1 #MyParm (A) DYNAMIC
  :
  WRITE *LENGTH(#MyParm)      /* ist 6
  #MyParm := '1234567'        /* verwendete Länge ist 7
  #MyParm := '12345678'       /* erweitert auf 8 Bytes
  EXPAND DYNAMIC VARIABLE #MyParm TO 10
  WRITE *LENGTH(#MyParm)      /* ist 8; zugewiesene Speichergröße
                               /* ist 10
END
```

3GL-Programmaufruf

Dynamische und große Variablen können mit dem CALL-Statement vernünftig verwendet werden, wenn die Option INTERFACE4 benutzt wird. Die Verwendung dieser Option führt zu einer Schnittstelle zu einem 3GL-Programm mit einer unterschiedlichen Parameter-Struktur. Für diese Einsatzmöglichkeit sind einige kleinere Änderungen im 3GL-Programm erforderlich; sie bietet aber im Vergleich zur älteren FINFO-Struktur erhebliche Vorteile, die im folgenden beschrieben sind. Weitere Informationen zur FINFO-Struktur siehe Call Interface4-Statement im *Natural Statements-Handbuch*.

- Keine Begrenzung der Anzahl der übergebenen Parameter (frühere Begrenzung 40).
- Keine Begrenzung der Größe der Parameter-Daten (frühere Begrenzung 64 KB pro Parameter).
- Komplette Parameter-Informationen können an das 3GL-Programm übergeben werden, einschließlich Array-Informationen.
Exportierte Funktionen stehen zur Verfügung, die einen sicheren Zugriff auf die Parameter-Daten ermöglichen (vorher mußten Sie darauf achten, Hauptspeicherplatz in Natural nicht zu überschreiben).

Vor dem Aufruf eines 3GL-Programms mit dynamischen Parametern muß sichergestellt werden, daß die erforderliche Puffergröße zugewiesen wird. Dies kann explizit mit dem EXPAND-Statement erfolgen.

Wenn ein initialisierter Puffer erforderlich ist, kann die dynamische Variable auf den Anfangswert und auf die erforderliche Größe gesetzt werden, wenn Sie das Statement MOVE ALL UNTIL verwenden. Natural stellt eine Reihe von Funktionen zur Verfügung, die es dem 3GL-Programm ermöglichen, Informationen über dynamische Parameter zu erhalten und die Länge zu ändern, wenn Parameter-Daten zurückgegeben werden.

Beispiel

```
MOVE ALL ' ' TO #MyDynText1 UNTIL 10000 /* ein Puffer der Länge 10000
                                         /* wird zugewiesen
                                         /* #MyDynText1 wird mit
                                         /* Leerstellen initialisiert
                                         /* *LENGTH wird auf 10000 gesetzt
CALL INTERFACE4 #3GLprog USING #MyDynText1
write *LENGTH(#MyDynText1) /* verwendete Länge kann sich im
                             /* 3GL-Programm geändert haben
:
```

Eine eingehendere Beschreibung entnehmen Sie dem CALL-Statement in der *Natural Statements-Dokumentation*.

Zugriff auf Arbeitsdateien mit großen und dynamischen Variablen

Große und dynamische Variablen können in Arbeitsdateien geschrieben oder aus Arbeitsdateien gelesen werden, und zwar mit den zwei Arbeitsdateitypen PORTABLE (portierbar) und UNFORMATTED (unformatiert). Für große/dynamische Variablen gibt es bei diesen Typen keine Speichergrößenbeschränkungen.

Die anderen Arbeitsdateitypen (ASCII, ASCII-COMPRESSED, ENTIRECONNECTION, SAG und TRANSFER) können keine dynamischen Variablen verarbeiten und führen zu einem Fehler. Große Variablen stellen für diese Arbeitsdateitypen kein Problem dar, es sei denn, die maximale Feld/Satzlänge wird überschritten (Feldlänge 255 für ENTIRECONNECTION und TRANSFER, Satzlänge 32767 für die anderen).

Beim Arbeitsdateityp PORTABLE werden die Feldinformationen in der Arbeitsdatei gespeichert. Die Längen der dynamischen Variablen werden bei einer READ-Leseoperation angepaßt, wenn die Feldlänge im Datensatz von der aktuellen Länge abweicht.

Der Arbeitsdateityp UNFORMATTED kann z.B. verwendet werden, um ein Video von einer Datenbank einzulesen und es in einer Datei zu speichern, die von anderen Utilities direkt abgespielt werden kann. Im WRITE WORK-Statement werden die Felder in ihrer Byte-Länge auf die Datei geschrieben. Alle Datentypen (DYNAMIC oder andere) werden gleich behandelt. Es werden keine Struktur-Informationen eingefügt. Beachten Sie, daß Natural einen Pufferungs-Mechanismus benutzt, so daß Sie davon ausgehen können, daß die Daten nur nach dem Statement CLOSE WORK komplett geschrieben werden. Dies ist besonders wichtig, wenn die Datei mit einer anderen Utility verarbeitet werden soll, während Natural läuft.

Beim READ WORK-Statement werden Felder fester Länge in ihrer ganzen Länge gelesen. Wenn das Dateiende (End-of-file) erreicht ist, wird der Rest des aktuellen Feldes mit Leerstellen gefüllt. Die folgenden Felder bleiben unverändert.

Im Falle von DYNAMIC-Datentypen wird der gesamte Rest der Datei gelesen, es sei denn, 1 GB wird überschritten. Wenn das Dateiende (End of file) erreicht wird, bleiben die restlichen Felder (Variablen) unverändert (normales Verhalten von Natural).

DDM-Generierung und -Bearbeitung für Spalten variabler Länge

In Abhängigkeit von den Datentypen wird das jeweils zutreffende Datenbank-Format A oder B generiert. Beim Datentyp VARCHAR der Datenbank wird die Natural-Länge der Spalte auf die Höchstlänge des Datentyps gesetzt, wie in der DBMS definiert. Im Falle von großen Datentypen wird das Schlüsselwort DYNAMIC an der Längensfeld-Stelle generiert.

Bei allen variierenden Längenspalten wird ein LINDICATOR-Feld L@<column-name> generiert. Für den Datentyp VARCHAR der Datenbank wird ein LINDICATOR-Feld mit Format/Länge I2 generiert. Bei großen Datentypen (siehe nachfolgende Liste) ist Format/ILänge I4.

In Zusammenhang mit dem Zugriff auf Datenbanken bietet die LINDICATOR-Verarbeitung die Möglichkeit, die Länge des zu lesenden Feldes zu erhalten oder die Länge des zu schreibenden Feldes zu setzen, und zwar unabhängig von einer definierten Pufferlänge (oder unabhängig von *LENGTH). Nach einer Retrieval-Suchfunktion wird *LENGTH gewöhnlich auf den entsprechenden Längenindikator-Wert gesetzt.

Beispiel-DDM

T	L	Name	F	Leng	S	D	Remark
:							
1	L@	PICTURE1	I	4			/* Längen-Indikator
1		PICTURE1	B	DYNAMIC			IMAGE
1	N@	PICTURE1	I	2			/* NULL-Indikator
1	L@	TEXT1	I	4			/* Längen-Indikator
1		TEXT1	A	DYNAMIC			TEXT
1	N@	TEXT1	I	2			/* NULL-Indikator
1	L@	DESCRIPTION	I	2			/* Längen-Indikator
1		DESCRIPTION	A	1000			VARCHAR(1000)
:							
:							
-----Extended Attributes-----							/* betrifft PICTURE1
Header	:	---					
Edit Mask	:	---					
Remarks	:	IMAGE					

Die generierten Formate sind Formate variierender Länge. Der Natural-Programmierer hat die Möglichkeit, die Definition von DYNAMIC auf eine Definition fester Länge zu ändern (extended field editing = erweitertes Editieren von Feldern) und z.B. die entsprechende DDM-Felddefinition für VARCHAR-Datentypen auf ein multiples Feld abzuändern (alte Generierung).

Beispiel

```

T  L  Name                F  Leng          S  D  Remark
:
  1  L@ PICTURE1          I   4
                                     /* Längen-Indikator
  1  PICTURE1             B 1000000000    IMAGE
  1  N@PICTURE1          I   2
                                     /* NULL-Indikator
  1  L@TEXT1              I   4
length indicator
  1  TEXT1                 A  5000          TEXT
  1  N@TEXT1              I   2
                                     /* NULL-Indikator
  1  L@DESCRIPTION        I   2
                                     /* Längen-Indikator
M  1  DESCRIPTION         A  100          VARCHAR(1000)
:
:
-----Extended Attributes-----
                                     /* betrifft PICTURE1

Header           :  ——
Edit Mask        :  ——
Remarks         :  IMAGE

```

Zugriff auf große Datenbank-Objekte

Um auf eine Datenbank mit großen Objekten (CLOBs oder BLOBs) zuzugreifen, ist ein DDM mit den entsprechend großen alphanumerischen oder binären Feldern erforderlich. Wenn eine feste Länge definiert wird, und wenn das große Objekte der Datenbank nicht in dieses Feld paßt, wird das große Objekt abgeschnitten.

Wenn der Programmierer nicht die definitive Länge des Datenbank-Objekts kennt, ist es sinnvoll, mit dynamischen Feldern zu arbeiten. So viele Neubelegungen (Reallokationen) wie erforderlich werden durchgeführt, um das Objekt komplett aufzunehmen, so daß kein Abschneiden nötig ist.

Beispiel-Programm

```

DEFINE DATA LOCAL
:
1 person VIEW OF xyz-person
  2 nachname
  2 vorname_1
  2 L@PICTURE1          /* I4 Längen-Indikator für PICTURE1
  2 PICTURE1           /* als dynamisch im DDM definiert
  2 TEXT1              /* als nicht-dynamisch im DDM definiert
:
END-DEFINE
:
SELECT * INTO VIEW person FROM xyz-person
                                /* PICTURE1 wird komplett gelesen
                                WHERE nachname = 'SMITH'
                                /* TEXT1 wird auf feste Länge 5000
                                /* abgeschnitten
:
  WRITE 'length of PICTURE1: ' L@PICTURE1
                                /* L-INDICATOR enthält die Länge von
:                                /* PICTURE1(= *LENGTH(PICTURE1)
/* etwas an PICTURE1 und TEXT1 verändern
:
L@PICTURE1 := 100000
INSERT INTO xyz-person (*) VALUES (VIEW person)
                                /* nur die ersten 100000 Bytes von
:                                /* PICTURE1 werden eingefügt.
:
END-SELECT

```

Wenn eine Format/Längen-Definition in dem View weggelassen wird, wird sie vom DDM übernommen.

Im *Reporting Mode* ist es jetzt möglich, eine beliebige Länge anzugeben, wenn das entsprechende DDM-Feld als DYNAMIC definiert ist. Das dynamische Feld wird auf ein Feld mit einer festen Pufferlänge abgebildet (Mapping). Umgekehrt ist dies nicht möglich.

DDM Format/Längen-Definition	VIEW Format /Längen-Definition	
(An)	– (An)	gültig gültig
	(Am) (A) DYNAMIC	nur gültig im Reporting Mode ungültig
(A) DYNAMIC	– (A) DYNAMIC (An) (Am / i : j)	gültig gültig nur gültig im Reporting Mode nur gültig im Reporting Mode

(Gleiches gilt für Variablen vom Format B.)

Parameter mit LINDICATOR-Klausel in SQL-Statements

Ist das LINDICATOR-Feld als I2 definiert, wird der SQL-Datentyp VARCHAR zum Senden oder Empfangen der entsprechenden Spalte verwendet. Ist die LINDICATOR-Hostvariable als I4 angegeben, wird ein großer Objekt-Datentyp (CLOB/BLOB) verwendet.

Wenn das Feld als DYNAMIC definiert ist, wird die Spalte in einer internen Schleife bis zu ihrer wirklichen Länge gelesen. Das LINDICATOR-Feld und *LENGTH werden auf diese Länge gesetzt. Im Falle eines Feldes fester Länge wird die Spalte bis zur definierten Länge gelesen. In beiden Fällen wird das Feld bis zu dem im LINDICATOR-Feld definierten Wert geschrieben.

Performance-Aspekte bei dynamischen Variablen

EXPAND und REDUCE

Der einer dynamischen Variablen zugewiesene Hauptspeicherplatz kann mittels des Statements REDUCE DYNAMIC VARIABLE auf eine angegebene Größe reduziert werden. Um einer Variablen eine angegebene Speichergröße neu zuzuweisen (Reallokierung), kann das EXPAND-Statement verwendet werden. (Wenn die Variable initialisiert werden sollte, verwenden Sie das Statement MOVE ALL UNTIL.)

Beispiel

```

DEFINE DATA LOCAL
:
#MyDynText1      (A)  DYNAMIC
# len            (I4)
:
END-DEFINE

#MyDynText1 := 'a'                /* verwendete Länge ist 1, Wert ist 'a';
                                /* zugewiesene Speichergröße ist noch 1

EXPAND DYNAMIC VARIABLE #MyDynText1 TO 100
                                /* verwendete Länge ist noch 1,
                                /* Wert ist 'a'; zugewiesene Größe ist 100
CALLNAT #subprog USING #MyDynText1
write *LENGTH(#MyDynText1)      /* verwendete Länge und zugewiesene Größe
                                /* können sich im Unterprogramm geändert haben

#len := *LENGTH(#MyDynText1)
REDUCE DYNAMIC VARIABLE #MyDynText1 TO #len
                                /* wenn die zugewiesene Größe die
                                /* verwendete Länge überschreitet, wird der
                                /* unbenutzte Hauptspeicherplatz freigegeben
:
REDUCE DYNAMIC VARIABLE #MyDynText1 TO 0
                                /* zugewiesenen Hauptspeicherplatz für
                                /* dynamische Variable freigeben

END

```

Regeln

- Verwenden sie dynamische Operanden, wo es sinnvoll erscheint.
- Verwenden Sie EXPAND, wenn die Obergrenze der Hauptspeicherbenutzung bekannt ist.
- Verwenden Sie REDUCE, wenn der dynamische Operand nicht mehr benötigt wird.

SYSTEMFUNKTIONEN

Für bestimmte Statements benutzte, in Natural eingebaute ("built-in") Funktionen sind in diesem Kapitel beschrieben.

Es behandelt die folgenden Themen:

- Allgemeines
- Alphabetische Liste der Systemfunktionen
- Mathematische Funktionen
- POS — Feldidentifikationsfunktion
- RET — Returncode-Funktion
- SORTKEY — Sortierschlüssel-Funktion

Allgemeines

Natural-Systemfunktionen können in einem MOVE-, COMPUTE-, DISPLAY-, PRINT- oder WRITE-Statement verwendet werden, das in einem der folgenden Statement-Blöcke steht:

- AT BREAK
- AT END OF DATA
- AT END OF PAGE

d.h. für alle FIND-, READ-, HISTOGRAM-, SORT- oder READ WORK FILE-Verarbeitungsschleifen.

Wenn eine Systemfunktion in einem AT END OF PAGE-Statement verwendet wird, muß das betreffende DISPLAY-Statement eine GIVE SYSTEM FUNCTIONS-Klausel enthalten.

Datensätze, die aufgrund einer WHERE-Klausel zurückgewiesen werden, werden von einer Systemfunktion nicht ausgewertet.

Wenn mittels Systemfunktionen Datenbankfelder ausgewertet werden, die aus FIND-, READ-, HISTOGRAM- oder SORT-Schleifen stammen, die auf verschiedenen Ebenen liegen, werden die Feldwerte jeweils entsprechend ihrer Position in der Verarbeitungsschleifen-Hierarchie verarbeitet. Werte einer äußeren Schleife werden zum Beispiel nur dann verarbeitet, wenn für diese Schleife neue Datenwerte erhalten werden.

Werden im *Reporting Mode* Systemfunktionen für Benutzervariablen ausgewertet, hängt ihre Verarbeitung davon ab, an welcher Stelle in der Verarbeitungsschleifen-Hierarchie die Variablen ursprünglich definiert wurden.

Wird eine Benutzervariable vor der ersten Verarbeitungsschleife definiert, wird sie für Systemfunktionen in der Schleife ausgewertet, in der das AT BREAK-, AT END OF DATA- oder AT END OF PAGE-Statement steht; wird eine Benutzervariable innerhalb einer Schleife definiert, wird sie in der gleichen Weise wie ein Datenbankfeld in der derselben Schleife behandelt.

Beim selektiven Einsatz von Systemfunktionen zur Auswertung von Benutzervariablen empfiehlt es sich, eine bestimmte Verarbeitungsschleife zu referenzieren (mittels Statement-Label oder Sourcecode-Zeilenummer), um genau festzulegen, in welcher Schleife der Wert der Benutzervariablen ausgewertet werden soll.

Systemfunktionen im SORT GIVE FUNCTIONS-Statement

Eine Systemfunktion kann auch referenziert werden, nachdem Sie in der GIVE FUNCTIONS-Klausel eines SORT-Statements ausgewertet wurde.

In diesem Fall muß dem Namen der Systemfunktion bei der Referenzierung ein Stern (*) vorangestellt werden.

Arithmetischer Überlauf bei AVER, NAVER, SUM oder TOTAL

Ein Feld, das Sie mit den Systemfunktionen AVER, NAVER, SUM oder TOTAL verwenden, muß lang genug sein (standardmäßig oder vom Benutzer definiert), um die Summe der Feldwerte aufzunehmen. Falls der addierte Wert die Länge des Feldes überschreitet, erhalten Sie eine Fehlermeldung. Normalerweise hat die Systemfunktion dieselbe Länge wie das angegebene Feld; ist diese Länge nicht ausreichend, dann verwenden Sie den Parameter NL, um die Ausgabelänge zu vergrößern:

SUM(Feld)(NL=nn)

Dadurch vergrößert sich nicht nur die Ausgabelänge sondern auch die interne Länge des Feldes.

Statement-Referenzierung (r)

Statement-Referenzierung kann auch auf Systemfunktionen angewandt werden. Weitere Einzelheiten entnehmen Sie dem Abschnitt **Statement-Referenzierung (r)** im Kapitel **Allgemeine Informationen** dieses Handbuchs.

Durch Verwendung eines Statement-Labels oder Angabe der Sourcecode-Zeilenummer (r) können Sie bestimmen, in welcher Verarbeitungsschleife die jeweilige Systemfunktion für das betreffende Feld ausgewertet werden soll.

ALPHABETISCHE LISTE DER SYSTEMFUNKTIONEN

- $AVER(r)(Feld)$
- $COUNT(r)(Feld)$
- $MAX(r)(Feld)$
- $MIN(r)(Feld)$
- $NAVER(r)(Feld)$
- $NCOUNT(r)(Feld)$
- $NMIN(r)(Feld)$
- $OLD(r)(Feld)$
- $SUM(r)(Feld)$
- $TOTAL(r)(Feld)$.

$AVER(r)(Feld)$

Format/Länge: Wie Feld. Ausnahme: bei einem Feld vom Format N hat $AVER(Feld)$ Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält den Durchschnittswert (Average) aller Werte des angegebenen Feldes. $AVER$ wird aktualisiert, wenn die Bedingung, unter der $AVER$ angefordert wurde, erfüllt ist.

$COUNT(r)(Feld)$

Format/Länge: P7.

Diese Systemfunktion zählt die Durchläufe einer Verarbeitungsschleife. Der Wert von $COUNT$ erhöht sich jedesmal um 1, wenn die Verarbeitungsschleife, in der sich $COUNT$ befindet, durchlaufen wird, und zwar unabhängig vom Wert des mit $COUNT$ angegebenen Feldes.

$MAX(r)(Feld)$

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den größten Wert des angegebenen Feldes. MAX wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich MAX befindet, ausgeführt wird.

$MIN(r)(Feld)$

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den kleinsten Wert des angegebenen Feldes. MIN wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich MIN befindet, ausgeführt wird.

$NAVER(r)(Feld)$

Format/Länge: Wie Feld. Ausnahme: bei einem Feld vom Format N hat NAVER(*Feld*)
Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält den Durchschnittswert (Average) aller Werte des angegebenen Feldes, wobei Nullwerte nicht berücksichtigt werden. NAVER wird aktualisiert, wenn die Bedingung, unter der NAVER angefordert wurde, erfüllt ist.

$NCOUNT(r)(Feld)$

Format/Länge: P7.

Diese Systemfunktion zählt die Durchläufe einer Verarbeitungsschleife. Der Wert von NCOUNT erhöht sich jedesmal um 1, wenn die Verarbeitungsschleife, in der sich NCOUNT befindet, durchlaufen wird, wobei Durchläufe, bei denen der Wert des angegebenen Feldes Null ist, nicht mitgezählt werden.

$NMIN(r)(Feld)$

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den kleinsten Wert des angegebenen Feldes, wobei Nullwerte nicht berücksichtigt werden. NMIN wird (falls erforderlich) jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich NMIN befindet, ausgeführt wird.

$OLD(r)(Feld)$

Format/Länge: Wie Feld.

Diese Systemfunktion enthält den "alten" Wert des angegebenen Feldes, d.h. den Wert, den das Feld vor einem in einer AT BREAK-Bedingung spezifizierten Gruppenwechsel (Wechsel des Feldwertes) bzw. vor einer Seitenende- oder Datenende-Bedingung (end of page, end of data) hatte.

$SUM(r)(Feld)$

Format/Länge: Wie Feld. Ausnahme: bei einem Feld vom Format N hat $SUM(Feld)$ Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält die Summe aller Werte des angegebenen Feldes. SUM wird jedesmal aktualisiert, wenn die Verarbeitungsschleife, in der sich SUM befindet, ausgeführt wird. SUM wird nach jedem AT BREAK-Gruppenwechsel wieder auf Null gesetzt, addiert also nur Werte zwischen zwei Gruppenwechseln.

$TOTAL(r)(Feld)$

Format/Länge: Wie Feld. Ausnahme: bei einem Feld vom Format N hat $TOTAL(Feld)$ Format P (mit derselben Länge wie das Feld).

Diese Systemfunktion enthält die Gesamtsumme aller Werte des angegebenen Feldes in allen offenen Verarbeitungsschleifen, in denen $TOTAL(Feld)$ vorkommt.

Systemfunktionen — Beispiel 2:

```

/* EXAMPLE 'ATBEX4': AT BREAK USING NATURAL SYSTEM FUNCTIONS
/*****
DEFINE DATA LOCAL
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 NAME
    2 CITY
    2 SALARY (2)
  1 #INC-SALARY (P11)
END-DEFINE
/*****
LIMIT 4
EMPLOOP. READ EMPLOY-VIEW BY CITY STARTING FROM 'ALBU'
  COMPUTE #INC-SALARY = SALARY (1) + SALARY (2)
  DISPLAY NAME CITY SALARY (1:2) 'CUMULATIVE' #INC-SALARY
  SKIP
  AT BREAK CITY
    WRITE NOTITLE
      'AVERAGE:' T*SALARY (1) AVER(SALARY(1)) /
      'AVERAGE CUMULATIVE:' T*#INC-SALARY
      AVER(EMPLOOP.) (#INC-SALARY)
  END-BREAK
END-READ
/*****
END

```

NAME	CITY	ANNUAL SALARY	CUMULATIVE
HAMMOND	ALBUQUERQUE	22000 20200	42200
ROLLING	ALBUQUERQUE	34000 31200	65200
FREEMAN	ALBUQUERQUE	34000 31200	65200
LINCOLN	ALBUQUERQUE	41000 37700	78700
AVERAGE:		32750	
AVERAGE CUMULATIVE:			62825

Systemfunktionen — Beispiel 3:

```

/* EXAMPLE 'AEDEX1S': AT END OF DATA (STRUCTURED MODE)
DEFINE DATA LOCAL
  1 EMPLOY-VIEW VIEW OF EMPLOYEES
    2 PERSONNEL-ID
    2 NAME
    2 FIRST-NAME
    2 SALARY (1)
    2 CURR-CODE (1)
END-DEFINE
/*
LIMIT 5
EMP. FIND EMPLOY-VIEW WITH CITY = 'STUTTGART'
  IF NO RECORDS FOUND
    ENTER
  END-NOREC
  DISPLAY PERSONNEL-ID NAME FIRST-NAME SALARY (1) CURR-CODE (1)
/*****
AT END OF DATA
  IF *COUNTER (EMP.) = 0
    WRITE 'NO RECORDS FOUND'
    ESCAPE BOTTOM
  END-IF
  WRITE NOTITLE / 'SALARY STATISTICS:'
    / 7X 'MAXIMUM:' MAX(SALARY(1)) CURR-CODE (1)
    / 7X 'MINIMUM:' MIN(SALARY(1)) CURR-CODE (1)
    / 7X 'AVERAGE:' AVER(SALARY(1)) CURR-CODE (1)
END-ENDDATA
/*****
END-FIND
END

```

PERSONNEL ID	NAME	FIRST-NAME	ANNUAL SALARY	CURRENCY CODE
11100328	BERGHAUS	ROSE	70800	DM
11100329	BARTHEL	PETER	42000	DM
11300313	AECKERLE	SUSANNE	55200	DM
11300316	KANTE	GABRIELE	61200	DM
11500304	KLUGE	ELKE	49200	DM
SALARY STATISTICS:				
	MAXIMUM:	70800	DM	
	MINIMUM:	42000	DM	
	AVERAGE:	55680	DM	

Systemfunktionen — Beispiel 4:

```

/* EXAMPLE 'AEPEX1S': AT END OF PAGE (STRUCTURED MODE)
/*****
DEFINE DATA LOCAL
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 PERSONNEL-ID
  2 NAME
  2 JOB-TITLE
  2 SALARY (1)
  2 CURR-CODE (1)
END-DEFINE
/*****
FORMAT PS=10
LIMIT 10
READ EMPLOY-VIEW BY PERSONNEL-ID FROM '20017000'
  DISPLAY NOTITLE GIVE SYSTEM FUNCTIONS
    NAME JOB-TITLE 'SALARY' SALARY(1) CURR-CODE (1)
/*****
  AT END OF PAGE
    WRITE / 28T 'AVERAGE SALARY: ...' AVER(SALARY(1)) CURR-CODE (1)
  END-ENDPAGE
/*****
END-READ
/*****
END

```

NAME	CURRENT POSITION	SALARY	CURRENCY CODE
CREMER	ANALYST	34000	USD
MARKUSH	TRAINEE	22000	USD
GEE	MANAGER	39500	USD
KUNEY	DBA	40200	USD
NEEDHAM	PROGRAMMER	32500	USD
JACKSON	PROGRAMMER	33000	USD
	AVERAGE SALARY: ...	33533	USD

Mathematische Funktionen

In logischen Bedingungen und den Arithmetik-Statements ADD, COMPUTE, DIVIDE, MULTIPLY und SUBTRACT können Sie die folgenden mathematischen Funktionen verwenden:

Funktion	Format/Länge	Ausgegebener Wert
ABS (<i>Feld</i>)	wie <i>Feld</i>	Absoluter Wert eines <i>Feldes</i> .
ATN (<i>Feld</i>)	F8 (*)	Arcustangens eines <i>Feldes</i> .
COS (<i>Feld</i>)	F8 (*)	Kosinus eines <i>Feldes</i> . Ist der Wert des <i>Feldes</i> größer oder gleich 10^{17} , ist COS (<i>Feld</i>) "1".
EXP (<i>Feld</i>)	F8 (*)	Potenz eines <i>Feldes</i> (exponential).
FRAC (<i>Feld</i>)	wie <i>Feld</i>	Bruchteil (hinter dem Komma) eines <i>Feldes</i> .
INT (<i>Feld</i>)	wie <i>Feld</i>	Ganzzahliger Teil eines <i>Feldes</i> (integer).
LOG (<i>Feld</i>)	F8 (*)	Natürlicher Logarithmus eines <i>Feldes</i> .
SGN (<i>Feld</i>)	wie <i>Feld</i>	Vorzeichen (sign) eines <i>Feldes</i> (-1, 0, +1).
SIN (<i>Feld</i>)	F8 (*)	Sinus eines <i>Feldes</i> . Wenn der Wert des <i>Feldes</i> größer oder gleich 10^{17} ist, ist SIN (<i>Feld</i>) "0".
SQRT (<i>Feld</i>)	(*)(**)	Quadratwurzel eines <i>Feldes</i> (square root). Ein negativer <i>Feldwert</i> wird wie ein positiver behandelt. Auf Großrechnern darf der <i>Feldwert</i> maximal 22 Stellen vor dem Komma haben.
TAN (<i>Feld</i>)	F8 (*)	Tangens eines <i>Feldes</i> . Wenn der Wert des <i>Feldes</i> größer oder gleich 10^{17} ist, ist TAN (<i>Feld</i>) "0".
VAL (<i>Feld</i>)	wie Zielfeld	Numerischer Wert eines alphanumerischen <i>Feldes</i> . Der Wert des alphanumerischen <i>Feldes</i> muß ein in alphanumerischer Form dargestellter numerischer Wert sein. Leerstellen vor und nach dem Komma im <i>Feld</i> werden ignoriert. Komma (Dezimalpunkt) und Vorzeichen werden mitverarbeitet. Wenn das Zielfeld nicht lang genug ist, werden Nachkommastellen abgeschnitten (vgl. Abschneiden und Runden von <i>Feldwerten</i> im Kapitel Allgemeine Informationen in diesem Handbuch).

- * Auf allen Plattformen, außer auf Großrechnern, werden diese Funktionen wie folgt ausgewertet: das Argument wird in Format/Länge F8 umgesetzt und dann zur Berechnung an das Betriebssystem übergeben; das Betriebssystem gibt das Ergebnis in Format/Länge F8 zurück, was dann in das Zielformat umgesetzt wird.
- ** Auf Großrechnern gilt:
 - Wenn *Feld* Format/Länge F4 hat, hat $\text{SQRT}(\text{Feld})$ auch Format/Länge F4;
 - wenn *Feld* Format/Länge F8 oder I hat, hat $\text{SQRT}(\text{Feld})$ Format/Länge F8;
 - wenn *Feld* Format N oder P hat, hat $\text{SQRT}(\text{Feld})$ Format/Länge $Nn.7$ bzw. $Pn.7$ (wobei n automatisch groß genug berechnet wird).

Ein mit einer mathematischen Funktion — außer VAL — verwendetes *Feld* kann eine Konstante oder ein Skalar sein und muß numerisches, gepackt numerisches, Ganzzahl- oder Gleitkomma-Format (N, P, I oder F) haben.

Ein mit der Funktion VAL verwendetes *Feld* kann eine Konstante, ein Skalar oder ein Array sein und muß alphanumerisches Format haben.

Beispiel für mathematische Funktionen:

```

/* EXAMPLE 'MATHEX': MATHEMATICAL FUNCTIONS
/*****
DEFINE DATA LOCAL
1 #A (N2.1) INIT <10>
1 #B (N2.1) INIT <-6.3>
1 #C (N2.1) INIT <0>
1 #LOGA (N2.6)
1 #SQRTA (N2.6)
1 #TANA (N2.6)
1 #ABS (N2.1)
1 #FRAC (N2.1)
1 #INT (N2.1)
1 #SGN (N1)
END-DEFINE
/*****
COMPUTE #LOGA = LOG(#A)
WRITE NOTITLE '=' #A 5X 'LOG' 40T #LOGA
/*****
COMPUTE #SQRTA = SQRT(#A)
WRITE      '=' #A 5X 'SQUARE ROOT' 40T #SQRTA
/*****
COMPUTE #TANA = TAN(#A)
WRITE      '=' #A 5X 'TANGENT' 40T #TANA
/*****
COMPUTE #ABS = ABS(#B)
WRITE      // '=' #B 5X 'ABSOLUTE' 40T #ABS
/*****
COMPUTE #FRAC = FRAC(#B)
WRITE      '=' #B 5X 'FRACTIONAL' 40T #FRAC
/*****
COMPUTE #INT = INT(#B)
WRITE      '=' #B 5X 'INTEGER' 40T #INT
/*****
COMPUTE #SGN = SGN(#A)
WRITE      // '=' #A 5X 'SIGN'      40T #SGN
/*****
COMPUTE #SGN = SGN(#B)
WRITE      '=' #B 5X 'SIGN'      40T #SGN
/*****
COMPUTE #SGN = SGN(#C)
WRITE      '=' #C 5X 'SIGN'      40T #SGN
/*****
END

```

#A:	10.0	LOG	2.302585
#A:	10.0	SQUARE ROOT	3.162277
#A:	10.0	TANGENT	0.648360
#B:	-6.3	ABSOLUTE	6.3
#B:	-6.3	FRACTIONAL	-0.3
#B:	-6.3	INTEGER	-6.0
#A:	10.0	SIGN	1
#B:	-6.3	SIGN	-1
#C:	0.0	SIGN	0

POS — Feldidentifikationsfunktion

Format/Länge: I4.

Die Systemfunktion POS(*Feldname*) enthält die interne Identifikation des Feldes, dessen Name mit der Systemfunktion angegeben wird.

POS(*Feldname*) identifiziert ein bestimmtes Feld, unabhängig von seiner Position in einer Map. Auch wenn sich die Reihenfolge und Anzahl der Felder in einer Map ändert, identifiziert POS(*Feldname*) nach wie vor eindeutig dasselbe Feld. Damit genügt zum Beispiel ein einziges REINPUT-Statement, um es von der Programmlogik abhängig zu machen, welches Feld MARKiert werden soll.

Beispiel:

```
DECIDE ON FIRST VALUE OF ...
  VALUE ...
    COMPUTE #FIELDX = POS(FIELD1)
  VALUE ...
    COMPUTE #FIELDX = POS(FIELD2)
  ...
END-DECIDE
...
REINPUT ... MARK #FIELDX
```

Wenn das mit POS angegebene Feld ein Array ist, muß eine bestimmte Ausprägung angegeben werden; zum Beispiel "POS(FELDX(5))". Auf einen Array-Bereich kann POS nicht angewendet werden.

POS und *CURS-FIELD

POS(*Feldname*) kann in Verbindung mit der Natural-Systemvariablen *CURS-FIELD (siehe Kapitel **Systemvariablen**) dazu verwendet werden, die Ausführung bestimmter Funktionen davon abhängig zu machen, in welchem Feld der Cursor zur Zeit steht.

*CURS-FIELD enthält die interne Identifikation des Feldes, in dem sich der Cursor zur Zeit befindet; *CURS-FIELD kann nicht alleine, sondern nur zusammen mit POS(*Feldname*) verwendet werden. Sie können beide zusammen dazu benutzen, zu prüfen, ob sich der Cursor gerade in einem bestimmten Feld befindet, und die weitere Verarbeitung von dieser Bedingung abhängig machen.

Beispiel:

```
IF *CURS-FIELD = POS(FIELDX)
  MOVE *CURS-FIELD TO #FIELDY
END-IF
...
REINPUT ... MARK #FIELDY
```

Anmerkung:

*Die Werte von *CURS-FIELD und POS(Feldname) dienen nur zur internen Identifikation der Felder und können nicht für arithmetische Operationen verwendet werden.*

Anmerkung für Natural RPC:

*Wenn *CURS-FIELD und POS(Feldname) sich auf eine Kontextvariable beziehen, können die daraus resultierenden Informationen nur innerhalb derselben Konversation verwendet werden.*

RET — Returncode-Funktion

Format/Länge: I4.

Die Systemfunktion `RET(Programmname)` kann dazu verwendet werden, den Returncode eines nicht in Natural geschriebenen Programms, das über ein `CALL`-Statement aufgerufen wurde, zu erhalten.

`RET(Programmname)` kann in einem `IF`-Statement sowie in den arithmetischen Statements `ADD`, `COMPUTE`, `DIVIDE`, `MULTIPLY` und `SUBTRACT` verwendet werden.

Beispiel:

```
DEFINE DATA LOCAL
1 #RETURN (I4)
...
END-DEFINE
...
...
CALL 'PROG1'
IF RET('PROG1') > #RETURN
    WRITE 'ERROR OCCURRED IN PROGRAM 1'
END-IF
...
```

SORTKEY — Sortierschlüssel-Funktion

Format/Länge: A253.

Es gibt in vielen Landessprachen Zeichen (oder Zeichenkombinationen), die von einem Sortierprogramm oder Datenbanksystem nicht in der richtigen alphabetischen Reihenfolge sortiert werden, da die Reihenfolge der Zeichen im vom Computer verwendeten Zeichensatz nicht immer der alphabetischen Reihenfolge der Zeichen entspricht.

Zum Beispiel wird der spanische Buchstabe “CH” in der Regel von einem Sortierprogramm bzw. Datenbanksystem wie zwei Buchstaben behandelt und zwischen “CG” und “CI” einsortiert — gehört aber eigentlich als eigener Buchstabe im spanischen Alphabet zwischen “C” und “D”.

Oder es kann sein, daß Kleinbuchstaben und Großbuchstaben entgegen Ihren Wünschen bei der Sortierreihenfolge nicht gleich behandelt werden, daß Ziffern vor Buchstaben sortiert werden (Sie aber wünschen, daß Buchstaben vor Ziffern sortiert werden) oder daß Sonderzeichen (z.B. Bindestriche in Doppelnamen) zu einer unerwünschten Sortierreihenfolge führen.

In solchen Fällen können Sie die Systemfunktion SORTKEY(*Zeichenkette*) benutzen, um “falsch sortierte” Zeichen (bzw. Zeichenkombinationen) in andere Zeichen (bzw. Zeichenkombinationen) umzusetzen, die dann vom Sortierprogramm oder Datenbanksystem alphabetisch “richtig sortiert” werden.

Die mit SORTKEY ermittelten Werte werden dann nur als Sortierkriterium verwendet, während die ursprünglichen Werte für die Interaktion mit dem Endbenutzer verwendet werden.

Sie können die SORTKEY-Funktion in einem COMPUTE-Statement sowie in einer logischen Bedingung als arithmetischen Operanden verwenden.

Als *Zeichenkette* können Sie eine alphanumerische Konstante oder Variable oder eine einzelne Ausprägung eines alphanumerischen Arrays angeben.

Wenn Sie die SORTKEY-Funktion in einem Natural-Programm angeben, wird der User-Exit NATUSK nn aufgerufen — wobei nn der aktuelle Sprachcode (d.h. der aktuelle Wert der Systemvariablen *LANGUAGE) ist.

Sie können diesen User-Exit in jeder Programmiersprache, die über eine Standard-CALL-Schnittstelle verfügt, schreiben. Die mit SORTKEY angegebene *Zeichenkette* wird an den User-Exit übergeben. Der User-Exit muß so programmiert werden, daß er “falsch sortierte” Zeichen in dieser Kette in entsprechende “richtig sortierte” Zeichen umsetzt. Die umgesetzte *Zeichenkette* wird dann vom Natural-Programm zur weiteren Verarbeitung verwendet.

Nähere Informationen zu dem User-Exit finden Sie in Ihrer *Natural Operations Documentation*.

Beispiel:

```

DEFINE DATA LOCAL
  1 CUST VIEW OF CUSTOMERFILE
    2 NAME
    2 SORTNAME
END-DEFINE
...
*LANGUAGE := 4
...
REPEAT
  INPUT NAME
  SORTNAME := SORTKEY(NAME)
  STORE CUST
  END TRANSACTION
  ...
END-REPEAT
...
READ CUST BY SORTNAME
  DISPLAY NAME
END-READ
...

```

Angenommen, im obigen Beispiel würden bei mehrmaliger Ausführung des INPUT-Statements nacheinander die Werte “Sanchez”, “Sandino” und “Sancinto” eingegeben.

Bei der Zuweisung von SORTKEY(NAME) zu SORTNAME würde der User-Exit NATUSK04 aufgerufen. Dieser müßte so programmiert werden, daß er zunächst alle Kleinbuchstaben in Großbuchstaben umsetzt und dann die Zeichenfolge “CH” in “CX” umsetzt — wobei *X* dem letzten Zeichen im verwendeten Zeichensatz entspricht, also hexadezimal H’FF’ (vorausgesetzt dieses letzte Zeichen ist kein druckbares Zeichen).

Es werden sowohl die “eentlichen” Namen (NAME) als auch die für die gewünschte Sortierung umgesetzten Namen (SORTNAME) gespeichert. Zum Lesen der Datei wird SORTNAME verwendet. Dann würden bei Ausführung des DISPLAY-Statements die Namen in der richtigen spanischen alphabetischen Reihenfolge ausgegeben:

```

Sancinto
Sanchez
Sandino

```

TERMINALKOMMANDOS

Dieses Kapitel beschreibt die Natural-Terminalkommandos.

Es ist in die folgenden drei Hauptabschnitte unterteilt:

- Benutzung von Terminalkommandos
- Nach Funktion eingeteilte Terminalkommandos
- Terminalkommando-Beschreibungen

Terminalkommandos – Übersicht

Terminalkommandos bieten eine Vielzahl unterschiedlichster Funktionen, die in den folgenden Abschnitten abgehandelt werden.

- Was sind Terminalkommandos?
- Eingabe eines Terminalkommandos
- Verwendung von Terminalkommandos in Programmen
- Verwendete Begriffe
- `??` – Hilfe für Terminalkommandos – nur auf Großrechnern
- Terminalkommando-Übersicht
- Tastenbelegungen für Terminalkommandos
- Nach Funktion eingeteilte Terminalkommandos

Was sind Terminalkommandos?

Mit Terminalkommandos können Sie im allgemeinen terminalbezogene Aktivitäten ausführen.

Das Steuerzeichen

Das erste Zeichen eines Terminalkommandos ist das Terminalkommando-Steuerzeichen, welches ein Terminalkommando als solches kennzeichnet. Das Standardsteuerzeichen ist das Prozentzeichen (%).

Ändern des Terminalkommando-Steuerzeichens

Sie können aber auch ein anderes Sonderzeichen als Steuerzeichen definieren, und zwar mit dem Session-Parameter CF (siehe Kapitel **Session-Parameter** in diesem Handbuch).

Wenn das Steuerzeichen geändert wird, werden alle Terminalkommandos, die einer Funktionstaste zugeordnet sind, automatisch entsprechend angepaßt.

Eingabe eines Terminalkommandos

Für die Verwendung von Terminalkommandos gelten folgende Regeln:

- Sie können ein Terminalkommando in jedem ungeschützten Feld auf dem Schirm eingeben (einschließlich der Meldungszeile, falls diese ungeschützt ist).
- Ein Terminalkommando muß in einem einzigen Feld eingegeben werden, mit Ausnahme des Steuerzeichens, das in ein vorangehendes Feld eingegeben werden kann.
- Es empfiehlt sich, nach einem Terminalkommando ein Leerzeichen einzugeben oder den übrigen Inhalt des Feldes, in das Sie das Kommando eingeben, zu löschen. Sonst kann es vorkommen, daß Natural den Feldinhalt irrtümlicherweise als Bestandteil des Terminalkommandos interpretiert.
- Sie können immer nur ein Terminalkommando zur selben Zeit ausführen. Geben Sie aber mehrere Terminalkommandos auf einmal ein, dann wird nur das erste ausgeführt; alle nachfolgenden werden ignoriert.
- Falsch eingegebene Terminalkommandos werden ignoriert, allerdings erhalten Sie keine entsprechende Fehlermeldung.
- Wenn Sie ein Terminalkommando und gleichzeitig in anderen Feldern auf dem Bildschirm Daten eingeben, wird nur das Terminalkommando ausgeführt; die Daten werden nicht verarbeitet.

Unter OpenVMS und UNIX

(Außer auf Natural-Menüscreens und Auswahl Fenstern) erscheint, sobald Sie das Steuerzeichen eingeben, ein Fenster, in dem Sie ein Terminalkommando eingeben können.

Verwendung von Terminalkommandos in Programmen

Terminalkommandos können mit einem SET CONTROL-Statement auch innerhalb eines Programms eingegeben werden; allerdings wird dann das Steuerzeichen weggelassen.

Verwendete Begriffe

In den Beschreibungen mehrerer Terminalkommandos werden die Begriffe “Bildschirm” und “Fenster” verwendet, und zwar mit folgenden Bedeutungen:

Bildschirm	Je nach dem Betriebssystem, unter dem Natural läuft, ist mit “Bildschirm” entweder der ganze Bildschirm als solcher bzw. das Betriebssystemfenster, in dem die Natural-Session läuft, bzw. das Natural-Hauptausgabefenster gemeint. Der Einfachheit halber wird jedoch in allen Fällen der Begriff “Bildschirm” (bzw. “Schirm”) verwendet.
Fenster	Hiermit ist immer das <i>Natural-Fenster</i> (wie unter Terminalkommando %W, Seite 352, erklärt) gemeint.

%? — Hilfe-Informationen für Terminalkommandos

%?

Dieses Kommando ist nur auf Großrechnern verfügbar.

Mit diesem Terminalkommando erhalten Sie Hilfe-Informationen für die Terminalkommandos von Natural.

Terminalkommando-Übersicht

Die folgende Tabelle bietet Ihnen einen Überblick über die zur Verfügung stehenden Terminalkommandos und ihre Funktionen.

Kommando	Funktion
%	Fortsetzungsanzeiger für INPUT-Statements in Batch-Programmen.
%% und %.	Unterbrechen der gerade aktiven Natural-Operation.
%*	Anzeige von Eingabezeichen unterdrücken. Im Batch-Betrieb: Ausdruck der nachfolgenden Eingabezeile unterdrücken.
%.P	Obersten Stack-Eintrag löschen.
%.S	Stack-Daten lesen ohne zu löschen.
%/	Erzwingen einer "End-of-File"-Bedingung bei einem INPUT-Statement im Batch-Betrieb.
%?	Aufrufen der Hilfe-Information zu Terminalkommandos.
%+ und %-	Ein- bzw. Ausschalten von Natural Connection.
%<TECH	Technische Informationen anzeigen.
%<TEST	Debugging-Funktion aufrufen.
%=	Zuordnen von Farben zu Feldern.
%A	Ausführen eines Recordings.
%B	Aktivieren/De-aktivieren des Recording-Prozesses.
%B=	Library für Recording.
%C	Kopieren des gegenwärtigen Bildschirminhalts in den Arbeitsbereich des Programm-Editors.
%CC, %CS	Kopieren von Daten in den Stack bzw. in die Systemvariable *COM.
%D	Aktivieren des Keyword/Delimiter-Modus.
%D=	Steuerung von "Outlining".
%E	Anzeige von mit NATPAGE aufgezeichneten Schirmen.
%E=	Aktivieren/Deaktivieren der Fehlerbehandlung.
%F	Aktivieren des Forms/Screens-Modus.
%F=	Rahmen-Zeichen für Bildschirmfenster.

Kommando	Funktion
%G	Setzen des Ausführmodus für Recordings.
%H	Erzeugen einer "Hardcopy"-Ausgabe.
%I	Aktuellen Schirm mit NATPAGE aufzeichnen.
%J	Aufrufen einer Helproutine.
%KN, %KO, %KS	Funktionstasten-Logik für Siemens-Terminals.
%Knn, %KPn	Simulieren von PF- und PA-Tasten.
%L	Umsetzung von Klein- in Großbuchstaben ausschalten.
%L=	Setzen des Sprachcodes.
%M	Steuerung der Meldungszeile.
%MSGSF	Anzeigeformat von Systemfehlermeldungen.
%N	Aktivieren des "Non-Conversational"-Modus.
%O	Beenden von NATPAGE.
%P	Aktivieren von NATPAGE für alle nachfolgenden Schirme.
%P=	CALL-Optionen.
%Q	Unterdrücken des Ausdrucks von Maps im Batch-Betrieb.
%QO	Pseudo-konversationale Ausgabe unterdrücken.
%QS	Gleichzeitige Ausgabe mehrerer Schirme.
%R	INPUT-Statement wiederholen.
%RM	Schreibschutz von lichtstift-sensitiven Feldern.
%RN	Bildschirmdaten-Komprimierung unterdrücken.
%RO	Aktivieren/Deaktivieren der Schirm-Optimierung.
%S	Fortsetzen von NATPAGE.
%T	Positionieren des Cursors am oberen Bildschirmrand.
%Tll/cc	Setzen der Cursor-Position.
%T+ und %T-	Positionieren des Cursors auf ein geschütztes Feld.
%T*	Positionieren des Cursors außerhalb des Fensters.
%T=	Aktivieren der terminalspezifischen Converter-Routine.
%TRE	Externe Trace-Funktion aktivieren/deaktivieren.

Kommando	Funktion
%TRI	Interne Trace-Funktion aktivieren/deaktivieren.
%U	Umsetzung von Klein- in Großbuchstaben einschalten.
%V	Steuerung des Print-Modus.
%W	Natural-Window-Steuerung.
%X	Steuerung der Statistikzeile/Infoline.
%Y	Steuerung der PF-Tastenleiste.
%Z	Löschen des Editor-Arbeitsbereichs.

Nach Funktion eingeteilte Terminalkommandos

Die folgende Tabelle liefert eine Übersicht über die nach Funktion eingeteilten Terminalkommandos.

- Umsetzen von Klein- in Großbuchstaben
- Kopieren/Löschen
- Sprache, Meldungen, Fehlerbehandlung
- Bildschirme, Terminal- und Fenster-Verarbeitung
- Farben, Outlining
- INPUT-Statement, Stack
- Recording (NATPAGE-Utility)
- Statistikzeile und Trace-Funktion
- Verschiedene
- Tastenbelegungen.

Umsetzen von Klein- in Großbuchstaben:	
%L	Umsetzung von Klein- in Großbuchstaben ausschalten.
%U	Umsetzung von Klein- in Großbuchstaben einschalten.
Kopieren, Löschen:	
%C	Aktuellen Bildschirm in Arbeitsbereich des Natural-Programm-Editors kopieren.
%CC	Daten in die Systemvariable *COM kopieren.
%CS	Daten in den Stack kopieren.
%Z	Arbeitsbereich des Natural-Programm-Editors löschen.
Sprache, Meldungen, Fehlerbehandlung:	
%E=	Fehlerbehandlung aktivieren/deaktivieren.
%L=	Sprachcode einstellen.
%M	Steuerung der Meldungszeile.
%MSGSF	Systemfehlermeldungen komplett anzeigen.
Bildschirme, Terminal- und Fenster-Verarbeitung:	
%F=	Rahmen-Zeichen für Fenster.
%K	Simulieren von PF- und PA-Tasten.
%Knn, %KPn	Simulieren von PF- und PA-Tasten.
%KN, %KO, %KS	Funktionstasten-Logik für Siemens-Terminals.
%N	Aktivieren des "Non- Conversational"-Modus.
%QS	Gleichzeitige Ausgabe mehrerer Schirme.
%RM	Schreibschutz von lichtstift-sensitiven Feldern.
%RN	Bildschirmdaten-Komprimierung unterdrücken.
%RO	Bildschirm-Optimierung ein-/ausschalten.
%T und %Tll/cc	Cursor-Position setzen.
%T*	Cursor außerhalb des Fensters plazieren.
%T+, %T-	Cursor in geschützte Felder plazieren.
%W	Window-Verarbeitung.
%Y	Steuerung der PF-Tastenzeilen.
%*	Anzeige von Eingabezeichen unterdrücken. Im Batch-Betrieb das Ausdrucken der nachfolgenden Eingabedatenzeile unterdrücken.

Farben, Outlining:	
%D=	“Outlining” steuern.
%=	Zuordnung von Farben zu Feldern.
INPUT-Statement, Stack:	
%	Fortsetzungsanzeiger für INPUT im Batch.
%D	Aktivieren des Keyword/Delimiter-Modus.
%F	Aktivieren des Forms/Screen-Modus.
%R	INPUT-Statement wiederholen.
%/	Erzwingen einer End-Of-File-Bedingung für INPUT im Batch-Betrieb.
%.P	Obersten Stack-Eintrag löschen.
%.S	Stack-Daten lesen ohne zu löschen.
Recording-Funktionen (NATPAGE-Utility auf Großrechnern):	
%A	Ausführen eines Recordings.
%B	Aktivieren/Deaktivieren des Recording-Prozesses.
%B=	Library für Recording.
%E	Mit NATPAGE aufgezeichnete Schirme anzeigen.
%G	Ausführmodus für Recording.
%I	Aktuellen Schirm mit NATPAGE aufzeichnen.
%O	Beenden von NATPAGE.
%P	NATPAGE für nachfolgende Schirme aktivieren.
%S	Fortsetzen von NATPAGE.
Statistikzeile und Trace-Funktion:	
%TRE	Externe Trace-Funktion aktivieren/deaktivieren.
%TRI	Interne Trace-Funktion aktivieren/deaktivieren.
%X	Steuerung der Statistikzeile/Infoline.
%<TECH	Technische Informationen anzeigen.
%<TEST	Debugging-Funktion aufrufen.

Verschiedene:	
%H	“Hardcopy”-Ausgabe.
%J	Helproutine aufrufen.
%P=	CALL-Optionen.
%Q	Map-Ausdruck im Batch-Betrieb unterdrücken.
%QO	Pseudo-konversationale Ausgabe unterdrücken.
%T=	Terminalspezifische Converter-Routine aktivieren.
%V	Steuerung des Print-Modus.
%?	Hilfe-Informationen für Terminalkommandos.
%%, %.	Unterbrechen der gerade aktiven Natural-Operation.
%+, %-	Natural Connection ein/ausschalten.
Tastenbelegungen:	
Taste(n)	Funktion
CLEAR	Unterbrechen der gerade aktiven Natural-Operation; Unterbrechen eines Recordings.
CTRL+D	Unterbrechen der gerade aktiven Natural-Operation.
RESET+ENTER	Beenden der aktiven Verarbeitungsschleife.

Terminalkommando-Tastenbelegungen

Die folgenden Themen werden hier erörtert:

Tasten	Funktion
CLEAR	Unterbrechen der gerade aktiven Natural-Operation; Unterbrechen eines Recordings.
CTRL+D	Unterbrechen der gerade aktiven Natural-Operation.
RESET+ENTER	Beenden der aktiven Verarbeitungsschleife.

Anmerkung:

*In einem Programm können Sie mit dem Statement `SET KEY` Terminalkommandos mit Funktionstasten belegen. Mit dem Systemkommando `KEY` können Sie auch Terminalkommandos mit Funktionstasten belegen (weitere Informationen siehe *Natural Statements-Handbuch*).*

CLEAR-Taste — Aktive Operation unterbrechen

Das Drücken der Taste CLEAR (bzw. LÖSCH) hat dieselbe Wirkung wie das Terminalkommando %%.
%%.

Natural für Großrechner:

Hier können Sie außerdem mit der CLEAR-Taste ein Recording unterbrechen, das im Film-Modus abläuft. Weitere Informationen zu Recordings finden Sie im Abschnitt **Debugging and Monitoring** in Ihrer *Natural Utilities Documentation*.

Natural für UNIX/OpenVMS:

Unter OpenVMS und UNIX können Sie anstelle der CLEAR-Taste die ESC-Taste verwenden — es sei denn, ESC ist in SAGtermcap eine andere Funktion zugewiesen.

CTRL+D-Tasten — Aktive Operation unterbrechen

Anmerkung:

Diese Funktion ist nur auf Großrechner-Terminals verfügbar.

Das Drücken der Tasten CTRL+D (bzw. STRG+D) hat dieselbe Wirkung wie das Terminalkommando %%.
%%.

RESET+ENTER-Tasten — Aktive Schleife abbrechen

Anmerkung:

Diese Funktion ist nur verfügbar unter COM-LETE und wenn Ihr Terminal über einen Local Controller angeschlossen ist.

Sie können die Verarbeitung einer gerade aktiven Verarbeitungsschleife abbrechen, indem Sie zuerst RESET (bzw. GRUNDSTELLUNG) und dann ENTER (bzw. EINGABE) drücken. Es muß sich um eine Schleife handeln, die einen Datenbankzugriff beinhaltet.

% — Fortsetzungsanzeiger für INPUT im Batch

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

%

Wenn “%” als letztes Nicht-Leerzeichen eines Datensatzes im Batch-Betrieb (gilt nur auf Großrechnern) verwendet wird, bewirkt es, daß der nächste Satz als Fortsetzungssatz behandelt wird.

Weitere Informationen siehe INPUT-Statement in der *Natural Statements-Dokumentation*.

%% und %. — Aktive Operation unterbrechen

Anmerkung:

Wenn der Profilparameter ESCAPE=OFF (der in Ihrer Natural Parameter Reference Documentation beschrieben ist) gesetzt ist, werden die Terminalkommandos “%%” und “%.” ignoriert.

Kommando-Syntax

%%

Diese Terminalkommandos können verwendet werden, um die aktuelle Operation zu unterbrechen.

%% im Online-Betrieb

Wenn Sie “%%” in irgendeinem Feld auf dem Schirm eingeben, wird das Natural-Programm, das gerade ausgeführt wird, sofort abgebrochen, und Sie gelangen wieder in den Kommandoeingabe-Modus.

Wenn Sie “%%” im Kommandoeingabe-Modus eingeben, wird die Natural-Session beendet (entspricht dem Systemkommando FIN).

“%%” hat folgende Auswirkungen:

- Der Inhalt des Natural-Stacks wird gelöscht.
- Eine logische Datenbank-Transaktion, die gerade ausgeführt wird, wird abgebrochen (BACKOUT).
- Das im Editor befindliche Source-Programm wird nicht beeinflusst und bleibt erhalten.

%% im Batch-Betrieb

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Im Batch-Betrieb können Sie mit “%%” Restart-Punkte in den Eingabedateien setzen und so die Synchronisation der Eingabedateien im Falle eines Fehlers sicherstellen.

Einfluß des Profilparameters CC

Kommando	Funktion
CC=ON	Wenn der Profilparameter CC=ON (vgl. Profile Parameters in der <i>Natural Parameter Reference Documentation</i>) gesetzt ist und bei der Kompilierung/Ausführung eines Natural-Programms im Batch-Betrieb ein Fehler auftritt, wird der Eingabedatenstrom für die Eingabedateien SYNIN und OBJIN bis zu der nächsten Zeile, die mit “%%” beginnt, gelöscht (wenn kein “%%” gefunden wird, wird er bis zum Dateiende gelöscht). Außerdem wird der Inhalt des Natural-Stacks gelöscht. Falls weitere Daten im Eingabestrom vorhanden sind, setzt Natural die Verarbeitung mit der Zeile nach “%%” fort.
CC=OFF	Bei CC=OFF wird “%%” in den Eingabedaten ignoriert.

%. im Online- und Batch-Betrieb

`%.`

Im Online-Betrieb entspricht “%.” dem Kommando “%%”, außer daß der Inhalt des Natural-Stacks nicht gelöscht wird.

Auf Großrechnern im Batch-Betrieb bewirkt “%.”, daß das Lesen der Eingabewerte für das gerade ausgeführte INPUT-Statement beendet wird.

%* — Anzeige von Eingabezeichen unterdrücken

Kommando-Syntax

```
%*
```

%* im Online-Betrieb

Die Verwendung dieses Terminalkommandos empfiehlt sich bei der Eingabe sensibler Daten (z.B. Paßwörter).

Dieses Terminalkommando bewirkt, daß sämtliche Daten, die auf dem aktuellen Schirm eingegeben werden, nicht angezeigt werden.

Wenn Sie %* mit einem SET CONTROL-Statement verwenden, werden die Eingaben sämtlicher Felder des nachfolgenden Schirms nicht angezeigt.

%* im Batch-Betrieb (nur auf Großrechnern)

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Im Batch-Betrieb wird mit %* das Ausdrucken der nachfolgenden Eingabedatenzeile unterdrückt. Die Zeile nach der Zeile, in der %* steht, wird nicht gedruckt. Damit können Sie zum Beispiel verhindern, daß Paßwörter mitausgedruckt werden.

Beispiel für %* im Batch-Logon nach Natural Security:

```
//CMSYNIN DD *  
%*  
SYSSEC, DBA, DBA  
...
```

Das Drucken aller Eingabedaten im Batch-Betrieb kann mit dem Profilparameter ECHO gesteuert werden (siehe **Profile Parameters** in der *Natural Parameter Reference Documentation*).

%P — Obersten Stack-Eintrag löschen

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

```
%P
```

Dieses Kommando löscht den obersten Eintrag aus dem Natural-Stack.

Nähere Informationen zum Stack finden Sie im *Natural Leitfaden zur Programmierung*.

%S — Stack-Daten lesen ohne zu löschen

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

```
%S
```

Normalerweise werden Kommandos/Daten vom Stack gelöscht, sobald sie aus dem Stack gelesen wurden.

Dieses Kommando bewirkt, daß mit dem nächsten INPUT-Statement der oberste Eintrag aus dem Stack gelesen wird, ohne gelöscht zu werden. Der Eintrag wird hierbei als Eingabedaten behandelt, ganz gleich ob es sich um Daten oder ein Kommando handelt.

Damit haben Sie die Möglichkeit, einen Stack-Eintrag zu prüfen und dann je nach seinem Inhalt zu entscheiden, ob sie ihn verarbeiten wollen oder nicht.

Den Inhalt des Stack können Sie auch mit der Systemvariablen *DATA prüfen.

Nähere Informationen zum Stack finden Sie im *Natural Leitfaden zur Programmierung*.

%/ — End-of-File

Anmerkung:

Dieses Kommando ist nur auf Großrechnern im Batch-Betrieb verfügbar.

Kommando-Syntax

```
%/
```

Dieses Terminalkommando erzwingt eine “End-of-File”-Bedingung, wenn es unmittelbar am Anfang eines im Batch-Betrieb von einem INPUT-Statement gelesenen Eingabedatensatzes steht.

%+ und %- — Natural Connection

Anmerkungen:

- ① Diese Kommandos sind unter Natural für Windows nicht verfügbar.
- ② Sie gelten nur, wenn Natural Connection installiert ist.

Kommando-Syntax

$$\% \left\{ \begin{array}{l} +[N] \\ - \end{array} [O] \right\}$$

Mit diesen Terminalkommandos schalten Sie Natural Connection ein bzw. aus.

%+	Mit diesem Kommando wird die Natural-Systemvariable *DEVICE auf "PC" gesetzt, was die Benutzung von Natural Connection ermöglicht.
%+N	Entspricht %+. Außerdem bewirkt dieses Kommando, daß bei einem Upload/Download von Daten keine Feldnamen übertragen werden. <i>Anmerkung:</i> <i>Dieses Kommando ist nur auf Großrechnern verfügbar.</i>
%+O	Entspricht %+. Außerdem bewirkt dieses Kommando, daß die neuen Funktionen von Natural Version 2.3 nicht unterstützt werden. Verwenden Sie Option "O", um Probleme zu vermeiden, wenn Sie mit älteren Versionen von Entire Connection arbeiten. <i>Anmerkung:</i> <i>Dieses Kommando ist nur auf Großrechnern verfügbar.</i>
%+NO %+ON	Kombiniert die Wirkungen von %+N und %+O. <i>Anmerkung:</i> <i>Diese Kommandos sind nur auf Großrechnern verfügbar.</i>
%-	Mit diesem Kommando wird die Natural-Systemvariable *DEVICE wieder auf den Wert gesetzt, den sie vor der Ausführung des Kommandos %+ hatte.

Weitere Informationen finden Sie in der *Natural Connection-Dokumentation*.

%<TECH — Technische Informationen anzeigen

Kommando-Syntax

```
%<TECH
```

Dieses Terminalkommando entspricht dem Systemkommando TECH.

%<TEST — Debugging-Funktion aufrufen

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

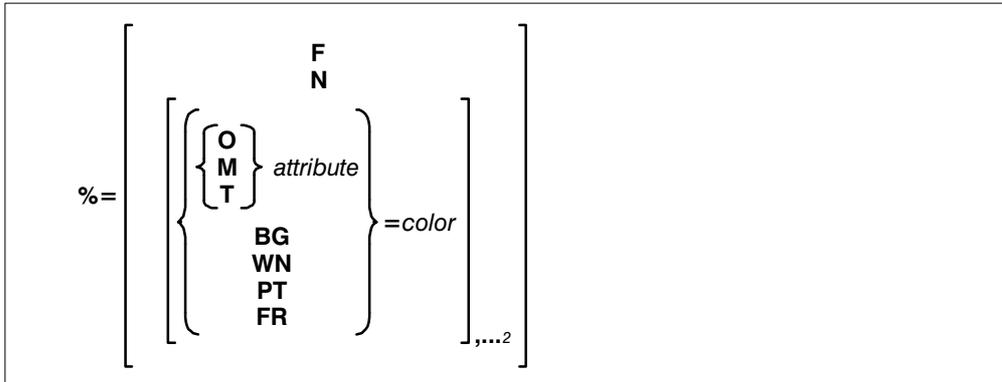
```
%<TEST
```

Mit diesem Terminalkommando rufen Sie die Debugging-Funktion auf. Es entspricht dem Systemkommando TEST (das in der *Natural Reference Documentation* beschrieben ist).

Nähere Informationen zur Debugging-Funktion finden Sie im Abschnitt **Natural Debugger** unter **Debugging and Monitoring** in Ihrer *Natural Utilities Documentation*.

%= — Zuordnen von Farben zu Feldern

Kommando-Syntax



Mit diesem Terminalkommando können Sie bestimmten Feldern bestimmte Farben zuweisen, und zwar für Programme, die ursprünglich ohne Berücksichtigung von Farbgebung geschrieben wurden.

Sie geben einen Feldtyp und/oder ein Feldattribut an sowie eine Farbe. Alle Felder dieses Typs/Attributs werden dann in dieser Farbe angezeigt.

Außerdem können Sie bestehende Farbzweisungen ändern, falls bereits vordefinierte Farbgebungen ungeeignet sind.

Darüber hinaus können Sie Farben dynamisch zuordnen (zum Beispiel bei der Erstellung einer Map).

Kommando	Funktion
Allgemeine Einstellungen:	
<i>Leerzeichen</i>	Bestehende Farbzuzuweisungen werden gelöscht.
F	Neu definierte Farbzuzuweisungen gelten statt denen des Programms.
N	Im Programm definierte Farbzuzuweisungen behalten ihre Gültigkeit.
Feldarten:	
O	Ausgabefelder (AD=O).
M	Eingabefelder; d.h. reine Eingabefelder (AD=A) sowie modifizierbare Felder (AD=M).
T	Textkonstanten.
Mögliche Feldattribute (<i>attribute</i>):	
B	Blinkend
C	Kursiv
D	Default, Standard
I	Intensiviert
U	Unterstrichen
V	Invers
Folgenden Bildschirmteilen kann ebenfalls eine Farbe zugewiesen werden:	
BG	Bildschirmhintergrund (background)
WN	Vordergrund (d.h. Felder, für die keine Farbe definiert ist)
PT	Standard-Seitenüberschrift (page title)
FR	Rahmen eines Bildschirmfensters
Mögliche Farben (<i>color</i>):	
BL	Blau
GR	Grün
NE	Neutral, weiß
PI	Pink
RE	Red, rot
TU	Türkis
YE	Yellow, gelb

Beispiel:**%=TI=RE,OB=YE**

Dieses Kommando bewirkt, daß alle intensiviert dargestellten Textfelder in Rot ausgegeben werden und alle blinkenden Ausgabefelder in Gelb.

%A — Ausführen eines Recordings

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

`%Aname`

Mit dem Terminalkommando `%Aname` führen Sie das unter *name* gespeicherte Recording aus. Voraussetzung ist, daß Ihre aktuelle Library diejenige ist, in der das Recording gespeichert ist.

Wenn Sie das Kommando `%Aname` während der Aufzeichnung einer Session eingeben, wird nicht das mit `%Aname` angegebene Recording ausgeführt, sondern das Kommando `%Aname` in die Source, die aufgezeichnet wird, eingefügt.

So können Sie aus einem Recording heraus ein anderes Recording ausführen und mehrere Recordings aneinanderreihen. Allerdings können Sie Recordings nicht schachteln; die Ausführung des Recordings, das das Kommando `%Aname` enthält, endet nach dem Kommando und wird nach der Ausführung von *name* nicht fortgesetzt.

Weitere Informationen zu Recordings finden Sie in der *Natural Utilities-Dokumentation für Großrechner*.

Nähere Informationen zu Recordings finden Sie im Abschnitt **Debugging and Monitoring** in Ihrer *Natural Utilities Documentation for Mainframes*.

%B — Aktivieren/Deaktivieren des Recording-Prozesses

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

```
%B[name]
```

Der Recording-Prozess wird mit dem Terminalkommando **%Bname** eingeschaltet und mit dem Terminalkommando **%B** wieder ausgeschaltet.

Kommando	Funktion
%Bname	Mit dem Terminalkommando %Bname schalten Sie den Recording-Vorgang ein. Alle nachfolgenden Aktionen werden aufgezeichnet. Die aufgezeichneten Daten werden unter dem angegebenen <i>Namen</i> gespeichert. Namen von Recordings müssen innerhalb einer Library eindeutig sein.
%B	Mit dem Terminalkommando %B schalten Sie den laufenden Recording-Vorgang aus. Die Aufzeichnung wird automatisch gespeichert, und Sie können sie beliebig oft wieder ausführen. Mit %B können Sie auch zusätzliche Aktionen in ein bestehendes Recording einfügen: nachdem Sie das Abspielen eines Recordings mit der LÖSCH-Taste unterbrochen haben, geben Sie das Kommando %B ein, und alle nachfolgenden Aktionen werden in das Recording eingefügt, solange bis Sie erneut %B eingeben. Das Abspielen des Recordings wird daraufhin fortgesetzt.

Weitere Informationen zu Recordings finden Sie im Abschnitt **Debugging and Monitoring** in der *Natural Utilities Documentation for Mainframes*.

%B= — Library für Recording

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

```
%B=library-name
```

Mit dem Terminalkommando `%B=library-name` geben Sie die Library an, in der alle anschließenden Aufzeichnungen gespeichert werden sollen.

Wenn Sie den Recording-Prozess aktivieren, ohne eine Library angegeben zu haben, wird das Recording in der Library gespeichert, deren Name dem Inhalt der Systemvariablen `*INIT-USER` zu Beginn des Aufzeichnungsvorgangs entspricht.

Wenn Sie während der Aufzeichnung einer Session die Library wechseln, bleibt die Library, in der die Aufzeichnung gespeichert wird, dieselbe (d.h. entweder die mit `%B=` angegebene oder die der Systemvariablen `*INIT-USER` entsprechende); dadurch ist es möglich, auch anwendungsübergreifende Aktionen aufzuzeichnen.

Weitere Informationen zu Recordings finden Sie im Abschnitt **Debugging and Monitoring** in der *Natural Utilities Documentation for Mainframes*.

%C — Seitenpuffer kopieren

Kommando-Syntax

%C

Mit diesem Terminalkommando kopieren Sie den Inhalt des Seitenpuffers (Page Buffer) in den Arbeitsbereich des Natural-Programm-Editors.

Die gegenwärtig von Natural angezeigte Seite wird in den Arbeitsbereich des Programm-Editors kopiert. Der Inhalt der Seite wird nach der letzten bereits im Editor befindlichen Sourcecode-Zeile eingefügt und kann dann mit dem Editor verändert werden.

Wollen Sie den bereits im Editor befindlichen Sourcecode vorher löschen, können Sie dazu das Terminalkommando %Z verwenden.

Anmerkungen:

- ① *%C sollte nicht in einer Editor-Session verwendet werden. Änderungen des Sourcecode-Arbeitsbereichs außerhalb des Editor-Inhalts werden vom Editor ignoriert.*
- ② *Der Seitenpuffer (die logische Natural-Ausgabe) ist nicht unbedingt mit dem am Bildschirm angezeigten Bildschirmpuffer identisch.*

Beispiel:

```
FOR I = 1 TO 10
  WRITE I
  SET CONTROL 'C'
END-FOR
END
```

%CS und %CC — Daten nach Stack/*COM kopieren

Kommando-Syntax

$$\%C \begin{Bmatrix} C \\ S \end{Bmatrix} \begin{Bmatrix} C \\ Lnn \end{Bmatrix} \begin{bmatrix} A \\ | \\ \text{farbe} \\ | \\ 1 \\ W \end{bmatrix} [s]$$

* **1** und **W** können nicht mit **Lnn** angegeben werden

Mit diesem Terminalkommando können Sie Teile des Bildschirms in den Natural-Stack (**%CS**) oder in die Systemvariable ***COM** (**%CC**) kopieren. Die geschützten Daten einer bestimmten Bildschirmzeile werden Feld für Feld kopiert (außer mit Option "A"; siehe im folgenden).

Der zweite Buchstabe des Kommandos bestimmt, wohin die Daten kopiert werden:

- **%CC...** schreibt die Daten in die Systemvariable ***COM**. Einzelheiten zu ***COM** finden Sie im Kapitel **Systemvariablen**.
- **%CS...** schreibt die Daten in den Natural-Stack. Die Daten werden oben auf dem Stack als Eingabedaten abgelegt (wie mit einem **STACK TOP DATA**-Statement).

Der dritte Buchstabe des Kommandos bestimmt, aus welcher Zeile Daten kopiert werden:

- **%CCC** und **%CSC** kopieren alle geschützten Daten aus der Zeile, in der sich der Cursor befindet, und zwar ab dem Feld, in dem sich der Cursor befindet.
- **%CCLnn** und **%CSLnn** kopieren alle geschützten Daten aus Zeile Nr. *nn*.

Darüber hinaus haben Sie folgende Optionen:

- **%C...A** kopiert eine komplette Zeile, d.h. nicht nur die geschützten Daten, sondern auch die modifizierbaren Felder; die Zeile wird nicht Feld für Feld kopiert, sondern als ganzes (einschließlich der Feldattribute).
- **%C...I** kopiert nur die intensiviert dargestellten Felder einer Zeile.
- **%C...farbe** kopiert nur die in der angegebenen *farbe* dargestellten Felder einer Zeile.
- **%C...C1** kopiert nur ein Feld, und zwar das, in dem sich der Cursor befindet (ungeachtet seiner Attribute). (**%C...Lnn1** ist nicht möglich.)
- **%C...CW** kopiert nur das Wort (begrenzt durch Leer- oder Sonderzeichen in einem Feld), auf dem der Cursor steht. (**%C...LnnW** ist nicht möglich.)
- **%C...S** bewirkt, daß Natural auf dem Schirm bleibt (“Stay-Option”), von dem die Daten kopiert werden, wenn das Kommando ausgeführt wird. Damit können Sie mehrere verschiedene Daten von einem Schirm kopieren, bevor Sie die Daten weiterverarbeiten.

Wenn Sie das Kommando **%C...** direkt eingeben (oder es einer PF-Taste zuweisen), bezieht es sich auf den *physischen Bildschirm* innerhalb des aktiven Fensters.

Ausnahme: Auf Großrechnern bezieht es sich auf den gesamten *physischen Bildschirm*; d.h. Sie können vom ganzen Schirm kopieren, ganz gleich ob die Daten, die Sie kopieren möchten, innerhalb oder außerhalb des aktiven Fensters sind.

Wenn Sie das Kommando über ein SET CONTROL-Statement (siehe *Natural Statements Documentation*) ausführen, bezieht es sich auf die von Natural erzeugte *logische Seite*. In Verbindung mit der “Stay“-Option ist es Ihnen dadurch möglich, sämtliche benötigten Daten von einer ganzen logischen Seite (die größer als der physische Schirm sein kann) zu kopieren, bevor Sie diese Daten weiterverarbeiten.

%D — Aktivieren des Keyword/Delimiter-Modus

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

```
%D
```

Mit diesem Terminalkommando aktivieren Sie den Keyword/Delimiter-Modus. Dieser Modus empfiehlt sich für die Batch-Verarbeitung.

Weitere Informationen siehe INPUT-Statement in der *Natural Statements-Dokumentation*.

%D= — “Outlining” steuern

Anmerkungen:

- ① *Dieses Kommando ist nur auf Großrechnern verfügbar.*
- ② *“Outlining” ist nur auf bestimmten Terminaltypen möglich, in der Regel auf solchen, die auch Doppelbyte-Zeichensätze (z.B. Kanji) unterstützen.*

Kommando-Syntax

```
%D=B {
      M
      I
      farbcode
      B
      R
      W
      } ...
```

Mit dem Terminalkommando %D=B steuern Sie das “Outlining”.

“Outlining” ist die Möglichkeit, bestimmte Felder auf dem Bildschirm “eingerahmt” (d.h. von einer Linie umgeben) anzuzeigen. Diese Form der Anzeige ist eine weitere Möglichkeit, dem Benutzer Länge und Position von Feldern auf dem Schirm deutlich zu machen.

Das Kommando %D=B bietet Ihnen folgende Möglichkeiten:

Kommando	Funktion
%D=BM	Dieses Kommando bewirkt, daß alle modifizierbaren Felder (AD=M oder AD=A) mit Rahmen angezeigt werden. Wenn dieses Kommando nochmals ausgeführt wird, wird diese Funktion wieder ausgeschaltet: mit anderen Worten, es erfolgt eine Rückkehr in den standardmäßigen Zustand.
%D=BI	Dieses Kommando bewirkt, daß alle intensivierten Felder (AD=I) mit Rahmen angezeigt werden.
%D=B<code>farbcode</code>	Dieses Kommando bewirkt, daß alle Felder der angegebenen Farbe mit Rahmen angezeigt werden. Gültige Farbcodes siehe Session-Parameter CD.
%D=BB	Dieses Kommando generiert große Rahmen (“big boxes”): wenn mehrere Felder mit identischen Anzeigecharakteristika untereinander angezeigt werden, wird ein einziger großer Rahmen um sie gezogen. Normalerweise (wenn %D=BB nicht angegeben wird) wird jedes Feld einzeln eingerahmt.
%D=BR	Dieses Kommando bewirkt, daß alle Rahmen-Anweisungen zurückgesetzt werden, d.h. kein Feld wird mit Rahmen angezeigt.
%D=BW	Dieses Kommando bewirkt, daß alle Fenster, die mit Rahmen angezeigt werden sollen (siehe DEFINE WINDOW-Statement in der <i>Natural Statements-Dokumentation</i>), mit Outlining-Rahmen statt mit Fenster-Rahmen angezeigt werden.

Wenn Outlining aktiv ist, werden Füllzeichen in modifizierbaren Feldern nicht angezeigt, da sie die gleiche Funktion erfüllen wie Outlining, nämlich Benutzern Position und Länge von Feldern deutlich zu machen.

%E — Mit NATPAGE aufgezeichnete Schirme anzeigen

Anmerkung:

Dieses Kommando ist unter Windows nicht verfügbar.

Kommando-Syntax

%E

Mit diesem Kommando können Sie sich Schirme, die mit der Utility NATPAGE (siehe *Natural Utilities Documentation*) aufgezeichnet wurden, anzeigen lassen.

Dieses Kommando unterbricht die normale Verarbeitung und aktiviert den Anzeigemodus von NATPAGE: Sie erhalten eine Liste aller mit NATPAGE (unter Verwendung der Terminalkommandos %I und %P) aufgezeichneten Schirme, jeweils mit Schirmnummer, Aufzeichnungszeitpunkt und Map-Name (falls der Schirm eine Map ist). Von der Liste können Sie die Schirme auswählen, die Sie sehen möchten. Sie können auf den angezeigten Schirmen keine Eingaben machen.

Wenn ein Schirm mit %I oder %P aufgezeichnet wird, wird eine Ecke des Schirms mit zwei Informationen überschrieben: der Uhrzeit, wann der Schirm aufgezeichnet wurde, sowie einer laufenden Nummer (die Schirme werden in der Reihenfolge, in der sie aufgezeichnet werden, fortlaufend nummeriert).

Natural für Großrechner:

Auf Großrechnern erhalten Sie im NATPAGE-Anzeigemodus ein Eingabefeld (CMD), in dem Sie folgende Blätterkommandos eingeben können:

Kommando	Funktion
TOP oder T	Zeigt den zuerst aufgezeichneten Schirm an.
BOT oder B	Zeigt den zuletzt aufgezeichneten Schirm an.
<i>nnn</i>	Zeigt den Schirm mit der Nummer <i>nnn</i> an.
+nnn	Blättert <i>nnn</i> Schirme vor.
-nnn	Blättert <i>nnn</i> Schirme zurück.
.	Beendet den Anzeigemodus.

Beim Versuch, zu einer nicht vorhandenen Schirmnummer zu blättern, wird — je nach Blätterraichtung — entweder der erste oder der letzte Schirm angezeigt.

Natural für OpenVMS und UNIX:

Unter OpenVMS und UNIX wird ein Fenster angezeigt, in dem Sie durch die Liste der aufgezeichneten Schirme blättern können. Mit EINGABE können Sie einen Schirm zur Anzeige auswählen. Mit ESC können Sie die Funktion verlassen.

Vgl. Terminalkommandos %P, %O, %S und %I.

%E= — Fehlerbehandlung ein-/ausschalten

Kommando-Syntax

$$\%E= \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \end{array} \right\}$$

Mit dem Terminalkommando %E=OFF schalten Sie eine etwaige Fehlertransaktion sowie die ON ERROR-Verarbeitung aus; mit %E=ON werden die Fehlertransaktion und die ON ERROR-Verarbeitung wieder eingeschaltet.

Kommando	Funktion
%E=OFF	Schaltet eine etwaige Fehlertransaktion (wie durch die Systemvariable *ERROR-TA identifiziert oder in Natural Security definiert) sowie ON ERROR-Verarbeitung aus; auftretende Fehler werden dann von der normalen Natural-Fehlerbehandlung verarbeitet. Dies kann dazu eingesetzt werden, einen Fehler in der Fehlerverarbeitung Ihrer Anwendung zu lokalisieren, falls es Ihnen aufgrund der Struktur der Anwendung mit verschiedenen Fehlerbehandlungsprozeduren auf verschiedenen Ebenen nicht möglich ist, herauszufinden, wo genau ein Fehler ursprünglich auftrat.
%E=ON	Mit dem Terminalkommando %E=ON schalten Sie dann Fehlertransaktion und ON ERROR-Verarbeitung wieder ein.

%F — Aktivieren des Forms/Screen-Modus

$$\%F$$

Dieses Kommando ist nur auf Großrechnern verfügbar.

Mit diesem Kommando aktivieren Sie den Forms/Screen-Modus. Weitere Informationen zu diesem Modus, siehe INPUT-Statement in der *Natural Statements-Dokumentation*.

%F — Aktivieren des Forms/Screen-Modus

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

```
%F
```

Mit diesem Terminalkommando können Sie den Forms/Screen-Modus aktivieren.

Weitere Informationen zum Forms/Screen-Modus entnehmen Sie dem INPUT-Statement (in der *Natural Statements-Dokumentation*).

%F= — Zeichen für Bildschirmfenster-Rahmen

Anmerkung:

Auf graphischen Benutzeroberflächen wird dieses Kommando ignoriert.

Kommando-Syntax

```
%F=chv
```

Mit diesem Terminalkommando können Sie die Zeichen definieren, die im Rahmen eines Bildschirmfensters verwendet werden sollen.

<i>c</i>	Das erste Zeichen wird für die vier <i>Ecken</i> des Rahmens verwendet.
<i>h</i>	Das zweite Zeichen wird für die <i>horizontalen</i> Linien des Rahmens (oberer und unterer Rand) verwendet.
<i>v</i>	Das dritte Zeichen wird für die <i>vertikalen</i> Linien des Rahmens (linker und rechter Rand) verwendet.

Beispiel: %F=+-! würde den Rahmen des Fensters beispielsweise so aussehen lassen:

```
+-----+
!                                     !
!                                     !
!                                     !
!                                     !
!                                     !
+-----+
```

Weitere Informationen zur Fensterverarbeitung siehe DEFINE WINDOW-Statement in der *Natural Statements-Dokumentation*.

%G — Ausführmodus für Recording

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

%G	[ON OFF]
----	---------------

Ein Recording kann in zwei Modi ausgeführt werden: im Hintergrundmodus oder im Filmmodus.

- **Hintergrundmodus**

Im Hintergrundmodus läuft das gesamte Recording unsichtbar ab; alle Aktionen des Recordings werden ausgeführt, ohne daß Ihnen während der Ausführung des Recordings irgendetwas am Bildschirm angezeigt wird. Außerdem können Sie ein im Hintergrundmodus ablaufendes Recording nicht unterbrechen (es sei denn, das Recording enthält das Terminalkommando %R).

- **Filmmodus**

Im Filmmodus wird ein Recording Schritt für Schritt ausgeführt, und alle Aktionen werden Ihnen angezeigt. Durch Drücken von EINGABE gelangen Sie zum jeweils nächsten Schritt. Im Filmmodus können Sie ein Recording außerdem durch Drücken von LÖSCH unterbrechen.

Standardmäßig läuft ein Recording im Hintergrundmodus ab.

%GON	Mit diesem Kommando schalten Sie den Filmmodus ein.
%GOFF	Mit diesem Kommando schalten Sie zurück zum Hintergrundmodus.
%G	Mit diesem Kommando wechseln Sie von einem Modus zum anderen.

Weitere Informationen zu Recordings finden Sie im Abschnitt **Debugging and Monitoring** in der *Natural Utilities Documentation for Mainframes*.

%H — “Hardcopy”-Ausgabe

Kommando-Syntax

$\%H[! \left[\begin{array}{c} \textit{destination} \\ , \\ \cdot \\ \# \textit{destination} \\ = [\textit{destination}] \\ - \end{array} \right]$
--

Dieses Kommando wird zum Drucken einer “Hardcopy” verwendet (falls diese Funktion implementiert ist).

Natural unter CMS:

Mit dem Kommando **%HL** wird eine Datei mit Namen NATURAL LISTING A angelegt.

Das Kommando **%H** erzeugt Ausgaben von Natural-Reports und Kommunikationsbildschirm-Layouts auf einem Hardcopy-Drucker. Standardmäßig bezieht sich ein %H-Kommando auf die aktuelle logische Seite (d.h. das aktuelle Bildschirmfenster ohne Meldungszeile, PF-Tastenleiste und Statistikzeile/Infoline). Sie können aber auch den aktuellen Bildschirm drucken (siehe **%H** und **%H.** unten).

Sie haben folgende Möglichkeiten:

Kommando	Funktion
%H	Sie erhalten in einem Fenster eine Liste aller verfügbaren Drucker, von der Sie den Drucker auswählen, auf dem die Hardcopy ausgedruckt werden soll. Natural für Großrechner: Sie erhalten kein Auswahlfenster; hier wird der mit dem Profilparameter HCDEST (vgl. <i>Natural Reference Documentation</i>) angegebene Drucker genommen. Natural für Windows: Es wird standardmäßig "LPT1" als Drucker genommen, wenn Sie keinen angeben.
%Hdestination	Die Hardcopy wird auf dem angegebenen Drucker (<i>destination</i>) ausgegeben. Die <i>destination</i> kann 1 bis 8 Zeichen lang sein.
%H!	<i>Anmerkung:</i> <i>Dieses Kommando ist nur unter UNIX und Windows verfügbar.</i> %H! gilt für die gerade auf dem Schirm angezeigte Seite.
%H!destination	<i>Anmerkung:</i> <i>Dieses Kommando ist nur unter UNIX und Windows verfügbar.</i> SET CONTROL 'H!destination' gilt jeweils nur für die aktuelle Seite (d.h. die Seite, die vor der Ausführung des SET CONTROL-Statements ausgegeben wurde).
%H,	Alle nachfolgenden %H-Kommandos beziehen sich auf den jeweils aktuellen <i>Bildschirminhalt</i> (nur möglich auf Großrechnern).
%H.	Alle nachfolgenden %H-Kommandos beziehen sich auf die jeweils aktuelle <i>logische Seite</i> (gilt standardmäßig).
%H=	Bei jedem nachfolgenden Drücken von EINGABE wird eine Hardcopy erzeugt (nochmalige Eingabe von %H= schaltet diesen Effekt wieder aus).
%H=destination	Mit %H=destination geben Sie an, auf welchem Hardcopy-Gerät <i>alle</i> logischen Ausgaben ausgegeben werden sollen. Diese Möglichkeit kann beispielsweise dazu genutzt werden, eine Abfolge von Ausgaben zu Verwaltungs-, Fehlerbeseitigungs- oder Schulungszwecken zu protokollieren.
%H#destination	Mit %H#destination können Sie die Hardcopy zu einem besonderen, mit einem DEFINE PRINTER-Statement definierten Ausgabeziel leiten, zum Beispiel in den Natural-Editorarbeitsbereich, nach Connect oder in die INFOLINE.
%H-	Hält die Ausgabe der Hardcopy sofort an. <i>Anmerkung:</i> <i>Wenn ein SET CONTROL 'H'-Statement ausgeführt wird, werden bereits in den Seitenpuffer geschriebene, aber noch nicht ausgegebene Daten nicht an den Drucker weitergeleitet. Um auch diese Daten zu drucken, müssen Sie vor dem SET CONTROL 'H'-Statement ein EJECT-Statement angeben.</i>

Grundsätzlich gilt ein %H-Kommando (ob irgendwo auf dem Bildschirm eingegeben oder mit einem SET CONTROL-Statement angegeben) entweder bis zur Ausführung des nächsten INPUT-Statements mit modifizierbaren Feldern oder bis zum Programmende. Wenn Sie allerdings %H in ein Eingabefeld eingeben, wird nur die aktuelle logische Seite gedruckt.

%I — Aktuellen Schirm mit NATPAGE aufzeichnen

Anmerkung:

Dieses Kommando ist unter Windows nicht verfügbar.

Kommando-Syntax

```
%I
```

Wenn Sie dieses Terminalkommando auf einem Schirm eingeben, wird der Schirm von der Utility NATPAGE (siehe *Natural Utilities Documentation*) aufgezeichnet.

Die maximale Anzahl von Schirmen, die mit NATPAGE aufgezeichnet werden können, bestimmen Sie mit dem Session-Parameter PD (siehe Kapitel **Session-Parameter**). Wenn diese Anzahl überschritten wird, überschreibt jeder weitere Schirm einen bereits aufgezeichneten, beginnend mit dem zuerst aufgezeichneten.

Um sich mit NATPAGE aufgezeichnete Schirme anzeigen zu lassen, verwenden Sie das Terminalkommando %E.

Vgl. Terminalkommandos %P, %O, %S und %E.

%J — Helproutine aufrufen

Kommando-Syntax

```
%Jhelproutine
```

Mit diesem Terminalkommando können Sie eine interaktive Helproutine aufrufen.

Wenn Sie %J eingeben, nachdem Sie die Ausführung eines Recordings unterbrochen haben (gilt nur auf Großrechnern), wird die Ausführung des Recordings nach Ausführung der Helproutine fortgesetzt (Einzelheiten zu Recordings finden Sie im Abschnitt **Debugging and Monitoring** in der *Natural Utilities Documentation for Mainframes*).

Wenn %J verwendet wird, wenn eine von einem Systemkommando aufgerufene Funktion aktiv ist, sucht Natural nach der angegebenen Helproutine, und zwar in der aktiven Library des Systemkommandos oder in einer Library, die als eine Steplib für das Systemkommando definiert wurde.

%KN/%KO/%KS — Siemens-Funktionstasten-Logik

Anmerkungen:

- ① Diese Kommandos sind nur für Großrechner verfügbar.
- ② Sie gelten nur für Siemens-Terminals.

Kommando-Syntax

$$\%K \begin{Bmatrix} N \\ O \\ S \end{Bmatrix} [N]$$

Die folgenden Kommandos sind verfügbar:

%KN	Mit diesem Kommando werden bei den Terminaltypen 8160, 974n, 9750 – 9755 die Literale “%K1” bis “%K20” auf die Tasten P1 bis P20 geladen; bei den Terminaltypen 9756, 9758, 976n werden die Funktionstastencodes “F1” bis “F20” auf die Tasten P1 bis P20 geladen.
%KO	Mit diesem Kommando werden die Literale “01” bis “20” sowie Funktionstastencode “F5” auf die Tasten P1 bis P20 geladen.
%KS	Mit diesem Kommando werden die Literale “A” bis “T” sowie Funktionstastencode “F5” auf die Tasten P1 bis P20 geladen.
%KNN %KON %KSN	Wenn Sie hinter dem jeweiligen Terminalkommando ein “N” angeben (d.h. %KNN, %KON oder KSN), wird nur der betreffende Funktionstastenmodus (KN, KO oder KS) aktiviert, aber es werden keine Werte auf die P-Tasten geladen.

Vgl. Abschnitt **Natural under BS2000/OSD** in der *Natural Operations Documentation for Mainframes*.

$\%K$ und $\%KP$ — Simulieren von PF- und PA-Tasten

Kommando-Syntax

$$\%K \left\{ \begin{array}{l} nn \\ Pn \end{array} \right\}$$

Diese Terminalkommandos können verwendet werden, um die Funktionstasten (PF, ENTER) und PA-Tasten zu simulieren.

Kommando	Funktion
$\%Knn$	<p>$\%Knn$ simuliert die Funktionstaste $PFnn$ (PF1 bis PF24). Dadurch können den PF-Tasten 1 bis 12 die Funktionen der PF-Tasten 13 bis 24 zugewiesen werden oder Funktionen von PF-Tasten, die auf der verwendeten Tastatur nicht vorhanden sind, aktiviert werden.</p> <p>Natural für Großrechner: Auf Großrechnern können dadurch außerdem PF-Tastenfunktionen im Batch-Betrieb verwendet werden.</p>
$\%K0$	$\%K0$ simuliert die ENTER- bzw. EINGABE-Taste.
$\%KPn$	$\%KPn$ simuliert die Funktionstaste PA_n (PA1 bis PA3) (siehe $\%Knn$).

%L - Keine Umsetzung von Klein- in Großbuchstaben

Kommando-Syntax

```
%L
```

Dieses Terminalkommando bewirkt, daß bei Eingabedaten Kleinbuchstaben nicht automatisch in Großbuchstaben umgesetzt werden.

Natural für Großrechner:

Auf Großrechnern sollten Sie sicherstellen, daß auch der verwendete TP-Monitor keine Umsetzung vornimmt, bevor Daten an Natural übergeben werden.

Vgl. Terminalkommando %U (Seite 350).

%L= — Sprachcode

Kommando-Syntax

```
%L=nn
```

Mit dem Terminalkommando %L=*nn* können Sie den Sprachcode *nn* setzen, der von Natural verwendet werden soll.

Eine Liste der möglichen Sprachcodes finden Sie bei der Beschreibung der Systemvariablen *LANGUAGE (siehe Kapitel **Systemvariablen**).

%M — Steuerung der Meldungszeile

Anmerkungen:

- ① Dieses Kommando wird im Batch-Betrieb ignoriert.
- ② Dieses Terminalkommando wirkt auch auf die NEXT- (bzw. MORE-Zeile) von Natural.

Kommando-Syntax

$$\%M \left[\begin{array}{c} T \\ B \\ [-]nn \\ \\ P \\ \\ \left. \begin{array}{c} BL \\ GR \\ NE \\ PI \\ RE \\ TU \\ YL \end{array} \right\} \end{array} \right]$$

Mit diesem Terminalkommando steuern Sie die Positionierung, den Schutz-Modus und die Farbe der Natural-Meldungszeile.

Positionierung der Meldungszeile

Kommando	Funktion
%MB	plaziert die Meldungszeile am unteren Bildschirmrand (B=Bottom).
%MT	plaziert die Meldungszeile am oberen Bildschirmrand (T=Top).
%M	bewirkt, daß die Position der Meldungszeile vom oberen zum unteren Bildschirmrand (oder umgekehrt) oder von Zeile <i>nn</i> zum unteren Bildschirmrand wechselt.
%Mnn	plaziert die Meldungszeile in die <i>nn</i> -te Zeile des Bildschirms.
%M- <i>nn</i>	plaziert die Meldungszeile in die <i>nn</i> -te Zeile von unten auf dem Bildschirm. Ist die angegebene Zeilennummer <i>nn</i> oder <i>-nn</i> außerhalb des aktuellen Bildschirms, wird die Meldungszeile nicht angezeigt.

Schutz der Meldungszeile

<code>%MP</code>	schaltet zwischen geschützter und ungeschützter Meldungszeile hin und her.
------------------	--

Farbe der Meldungszeile

<code>%M=farbcode</code>	zeigt die Meldungszeile in der angegebenen Farbe an (zur Bedeutung der einzelnen Farbcodes siehe Session-Parameter CD).
--------------------------	--

%MSGSF — Anzeigeformat von Systemfehlermeldungen

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

$$\%MSGSF = \left\{ \begin{array}{c} \text{ON} \\ + \\ \text{T} \\ \text{OFF} \\ - \\ \text{F} \end{array} \right\}$$

Standardmäßig besteht eine Natural-Systemfehlermeldung aus dem Namen des Programms und der Nummer der Zeile, das/die den Fehler verursacht hat, gefolgt von dem eigentlichen Text der Meldung. Je nach Größe des Fensters, in dem die Meldung angezeigt wird, kann es sein, daß der eigentliche Text nicht vollständig angezeigt, sondern abgeschnitten wird. Mit dem Terminalkommando %MSGSF können Sie dies vermeiden.

Die folgenden Kommando-Optionen sind verfügbar:

%MSGSF=ON %MSGSF=+ %MSGSF=T	Systemfehlermeldungen werden komplett angezeigt; d.h. Programmname, Zeilennummer und eigentlicher Meldungstext werden angezeigt (dies ist die Standardeinstellung).
%MSGSF=OFF %MSGSF=- %MSGSF=F	Systemfehlermeldungen werden in Kurzform angezeigt; d.h. es wird nur der eigentliche Meldungstext angezeigt (aber nicht der Programmname und die Zeilennummer).

Anmerkungen:

- 1 Anstelle von "ON" können Sie auch "+" oder "T" (true) angeben; anstelle von "OFF" können Sie auch "-" oder "F" (false) angeben.
- 2 Die Anzeigeform von Systemfehlermeldungen kann auch mit dem Profilparameter MSGSF gesteuert werden. Siehe den Abschnitt **Profile Parameters** in der *Natural Parameter Reference Documentation*.

%N — Aktivieren des “Non-Conversational”-Modus

Kommando-Syntax

```
%N
```

Dieses Terminalkommando wird mit einem SET CONTROL-Statement verwendet und bewirkt, daß der nächste Schirm angezeigt wird und die Verarbeitung anschließend sofort fortgesetzt wird, ohne daß es hierzu einer Benutzerreaktion bedarf; d.h. nachdem der Schirm angezeigt wurde, wird die Verarbeitung sofort fortgesetzt, ohne daß auf eine Benutzereingabe gewartet wird.

Dieses Kommando kann dazu verwendet werden, Informationen über den Stand des Programmablaufs an den Benutzer zu schicken.

Das Kommando bezieht sich jeweils nur auf den nachfolgenden Ausgabeschirm (d.h. den Schirm, der ausgegeben wird, wenn der nächste I/O ausgeführt wird).

Unter IMS/TM:

Unter IMS/TM gilt dieses Terminalkommando nicht, wenn eine IMS-“Scratchpad Area” verwendet wird.

%O — Beenden von NATPAGE

Anmerkung:

Dieses Kommando ist unter Windows nicht verfügbar.

Kommando-Syntax

%O

Mit diesem Terminalkommando beenden Sie das Aufzeichnen von Bildschirmen mit NATPAGE (das mit dem Terminalkommando %P aktiviert wurde). Weitere Informationen zur NATPAGE-Utility entnehmen Sie der *Natural Utilities Documentation*.

Der aktuelle Schirm wird noch aufgezeichnet. Alle seit dem letzten %P-Kommando aufgezeichneten Schirme bleiben gespeichert.

Vgl. Terminalkommandos %E, %I, %P und %S.

%P – NATPAGE für nachfolgende Schirme aktivieren

Anmerkung:

Dieses Kommando ist unter Windows nicht verfügbar.

Kommando-Syntax

%P

Mit diesem Terminalkommando aktivieren Sie die Utility NATPAGE, um den aktuellen Schirm und alle nachfolgenden Schirme aufzuzeichnen. Weitere Informationen zur NATPAGE-Utility entnehmen Sie der *Natural Utilities Documentation*.

Die maximale Anzahl von Schirmen, die mit NATPAGE aufgezeichnet werden können, bestimmen Sie mit dem Session-Parameter PD. Wenn diese Anzahl überschritten wird, überschreibt jeder weitere Schirm einen bereits aufgezeichneten, beginnend mit dem zuerst aufgezeichneten.

Alle Schirme, die mit vorhergehenden %P- bzw. %I-Kommandos aufgezeichnet wurden, werden gelöscht, wenn Sie ein %P-Kommando ausführen.

Um sich mit NATPAGE aufgezeichnete Schirme anzeigen zu lassen, verwenden Sie das Terminalkommando %E.

Vgl. Terminalkommandos %E, %I, %O und %S.

%P= — CALL-Optionen

Anmerkung:

Die folgenden Kommandos gelten nur auf Großrechnern.

Kommando-Syntax

$$\%P= \left\{ \begin{array}{c} S \\ V \\ C \\ L \end{array} \right\}$$

Mit den Kommandos %P=S, %P=V und %P=C können Sie spezielle Optionen setzen, die gelten, wenn unter CICS ein Natural-Programm (über ein CALL-Statement) ein Nicht-Natural-Programm aufruft. In allen anderen Umgebungen werden diese Kommandos ignoriert.

Zu Einzelheiten über den Aufruf von Nicht-Natural-Programmen siehe CALL-Statement in der *Natural Statements-Dokumentation*.

Ein %P=-Kommando gilt nur für den jeweils nachfolgenden Aufruf.

%P=S — Standard-Linkage für Aufruf

Anmerkung:

Dieses Kommando gilt nur unter CICS.

Wenn ein Natural-Programm unter CICS ein Nicht-Natural-Programm aufruft, geschieht der Aufruf normalerweise über eine "EXEC CICS LINK"-Anforderung.

Wenn für den Aufruf stattdessen Standard Linkage verwendet werden soll, geben Sie das Terminalkommando %P=S ein. Voraussetzung ist, daß das aufgerufene Programm den Standard-Linkage-Konventionen mit Standard-Registerverwendung entspricht.

%P=V — Ausspeichern bei Aufruf

Anmerkung:

Dieses Kommando gilt nur unter CICS.

Wenn ein Natural-Programm ein Nicht-Natural-Programm aufruft und das aufgerufene Programm im Dialog einen Terminal-I/O ausführt, ist der Natural-Thread normalerweise solange blockiert, bis der Benutzer eine Eingabe gemacht hat.

Unter CICS können Sie mit %P=V vermeiden, daß der Natural-Thread blockiert ist: wenn Sie dieses Terminalkommando angeben, werden die vom Natural-Programm an das aufgerufene Programm übergebenen Parameterdaten aus dem Thread herauskopiert, und der Thread wird vor dem Aufruf ausgespeichert. Der Thread steht dann für andere Benutzer zur Verfügung.

Wenn das aufgerufene Programm die Kontrolle wieder an das aufrufende Natural-Programm zurückgibt, wird der Thread wieder eingespeichert, der (modifizierte) Datenbereich wird in den Thread kopiert, und Natural setzt die Verarbeitung fort.

Anmerkung:

Nur die im CALL-Statement angegebenen Parameter werden aus dem Thread heraus- und wieder hineinkopiert.

%P=C — Übergeben von Parameterwerten statt -adresse

Anmerkung:

Dieses Kommando gilt nur unter CICS.

Ruft ein Natural-Programm unter CICS ein Nicht-Natural-Programm auf, wird normalerweise die Adresse der CALL-Parameter-Adressenliste in der COMMAREA übergeben. Wenn Sie statt der Adresse der Parameteradressenliste die Parameterwerte selbst in der COMMAREA übergeben möchten, führen Sie vor dem Aufruf das Terminalkommando %P=C aus.

Dadurch ist zum Beispiel DPL für aufgerufene CICS-Programme möglich: der Aufruf eines CICS-Programms, das sich in einer anderen CICS-Region befindet, ist nur mit %P=C möglich; denn da die "aufgerufene" Region nicht auf Adressen in der "aufrufenden" Region zugreifen kann, müssen stattdessen die Parameterwerte übergeben werden.

Wenn %P=C verwendet wird, werden keine Parameter in der TWA übergeben, sondern nur Parameterwerte in der CICS COMMAREA. Alle Parameter der CALL-Parameterliste werden unmittelbar aufeinanderfolgend kopiert, unabhängig von ihrer Ausrichtung. Die sich daraus ergebende COMMAREA-Länge entspricht der Summe der einzelnen Parameterlängen (was bei Bestimmung der Anzahl der zu übergebenden Array-Ausprägungen zu berücksichtigen ist). Bei der Rückkehr vom aufgerufenen Programm werden die Parameter zurückkopiert.

Wenn sich überlappende Felder übergeben werden oder dasselbe Feld mehrmals übergeben wird, sollten diese Felder für das aufgerufene Programm schreibgeschützt werden; andernfalls kann es bei der Rückgabe von Parameterwerten an das aufrufende Programm zu unvorhersehbaren Ergebnissen kommen.

Bei %P=C gilt die Einschränkung, daß Gruppen-Arrays nicht übergeben werden können:

```
01 #GROUP (2)
   02 #FIELD1 (A1)
   02 #FIELD2 (P7)
```

Übergeben Sie sie entweder als einzelne Arrays:

```
01 #GROUP
   02 #FIELD1 (A1/2)
   02 #FIELD2 (P7/2)
```

Oder redefinieren Sie sie:

```
01 #GROUP
01 REDEFINE #GROUP
   02 #ARRAY (A1/10)
```

und geben den Array-Namen im CALL-Statement an.

Anmerkung:

*Wenn Sie vor einem Aufruf %P=S **und** %P=C ausführen, wird %P=C ignoriert.*

%P=L — Aufruf des dynamischem Hauptprogramms LE/370

Anmerkung:

Dieses Kommando gilt nur, wenn bei der Installation von Natural die Option zur Unterstützung von "IBM Language Environment (LE/370)"-Aufrufkonventionen gesetzt wurde.

Dieses Kommando bewirkt, daß nach dem Aufruf eines dynamischen Hauptprogramms LE/370 die Kontrolle wieder an Natural zurückgegeben wird.

Wenn ein dynamisches Hauptprogramm LE/370 aufgerufen wird, wird standardmäßig die Kontrolle nach Ausführung des aufgerufenen Programms nicht an Natural zurückgegeben. Damit nach der Ausführung des aufgerufenen Programms die Kontrolle an Natural zurückgegeben wird, müssen Sie vor dem Programmaufruf %P=L verwenden.

Informationen zur Unterstützung von LE/370-Aufrufkonventionen finden Sie in der *Natural Operations Documentation for Mainframes*.

%Q – Map-Ausdruck im Batch-Betrieb unterdrücken

Kommando-Syntax

```
%Q
```

Im Online-Betrieb wird %Q ignoriert.

Im Batch-Betrieb (gilt nur auf Großrechnern) bewirkt %Q, daß Maps oder Schirme, die über INPUT-Statements erzeugt werden, nicht mit ausgegeben werden.

SET CONTROL 'Q' (im Online- oder Batch-Betrieb) bewirkt, daß das nächste INPUT-Statement *nicht* verarbeitet wird. Dies können Sie beispielsweise einsetzen, wenn nach Beendigung einer Helproutine die Verarbeitung bei der Rückkehr von der Helproutine zur Map sofort fortgesetzt werden soll, ohne daß der Benutzer EINGABE drücken muß.

%QO — Pseudo-konversationale Ausgabe unterdrücken

Anmerkungen:

- ① *Dieses Kommando ist nur auf Großrechnern verfügbar.*
- ② *Dieses Kommando gilt nur unter CICS.*

Kommando-Syntax

```
%QO
```

Innerhalb einer Natural-Session, die unter CICS im Pseudo-Dialogbetrieb (pseudo-conversational mode) abläuft, können Sie mit dem Statement CALL 'CMTASK' eine andere CICS-Transaktion starten. Damit CICS die Transaktion starten kann, ist allerdings nach dem CALL-Statement ein Terminal-I/O, d.h. ein INPUT-Statement, erforderlich.

Um die Bildschirmausgabe dieses INPUT-Statements (die ohnehin sofort von der gestarteten Transaktion überschrieben würde) zu unterdrücken, führen Sie vor dem INPUT-Statement das Statement SET CONTROL 'QO' aus.

%QS — Gleichzeitige Ausgabe mehrerer Schirme

Kommando-Syntax

```
%QS
```

Mit diesem Kommando können Sie mehrere Ausgabeschirme gleichzeitig anzeigen.

%QS bewirkt, daß der nächste Bildschirm-I/O nicht ausgeführt wird. Der betreffende Ausgabeschirm wird intern zwischengespeichert und erst beim darauffolgenden I/O zusammen mit dem nächsten Schirm ausgegeben.

%QS ist also nur sinnvoll, wenn der zweite Ausgabeschirm ein Fenster ist, d.h. den ersten, mit %QS unterdrückten Schirm nicht vollständig überlagert.

Beispiel:

So können Sie zum Beispiel die Ausgabe eines Schirmes A mit %QS unterdrücken; der nächste Schirm B ist ein Fenster, das Schirm A teilweise überlagert (beispielsweise ein Hilfefenster zu einem der Felder auf Schirm A); mit dem nächsten Bildschirm-I/O wird dann das Fenster B zusammen mit dem "darunterliegenden" Schirm A gleichzeitig angezeigt.

Ein %QS-Kommando gilt nur für den jeweils nachfolgenden Schirm.

Anmerkung:

Da sich durch %QS die Anzahl der Bildschirm-I/Os reduziert, spart %QS außerdem Verarbeitungszeit.

%R — INPUT-Statement wiederholen

Kommando-Syntax

```
%R
```

Dieses Terminalkommando bewirkt, daß ein INPUT-Statement wiederholt und der entsprechende Ausgabeschirm erneut generiert wird. Alle seit dem Beginn des INPUT-Statements generierten Ausgabedaten werden reproduziert.

%R in einem Recording

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Durch Aufzeichnen des Terminalkommandos %R können Sie einen einzelnen Schritt in einem Recording während der Ausführung des Recordings verändern.

Sie können %R auch dazu benutzen, Eingabedaten in einem Recording, das ausgeführt wird, zu überschreiben.

Näheres hierzu siehe den Abschnitt **Debugging and Monitoring** in der *Natural Utilities Documentation for Mainframes*.

%RM – Schreibschutz von lichtstift-sensitiven Feldern

Anmerkung:

Die folgenden Kommandos gelten nur auf Großrechnern.

Kommando-Syntax

%RM

Mit diesem Kommando machen Sie alle lichtstift-sensitiven Felder auf dem Schirm schreibgeschützt, d.h. der Benutzer kann sie mit dem Lichtstift auswählen, aber ihren Inhalt nicht überschreiben. Um den Schreibschutz wieder auszuschalten, geben Sie das Kommando noch einmal ein.

Um lichtstift-sensitiv zu sein, muß ein Feld intensiviert (AD=I) oder blinkend (AD=B) angezeigt werden, und das erste Zeichen im Feld muß eines der Lichtstift-Funktionszeichen ("?", ">", "&" oder ein Leerzeichen) sein. Wenn ein Feld mit dem Lichtstift ausgewählt wird, ändert sich das Funktionszeichen; Sie können also die Verarbeitung von mit Lichtstift ausgewählten Feldern vom Wert des Funktionszeichens abhängig machen.

Siehe auch den Abschnitt **Light Pen Support** in der *Natural Operations for Mainframes Documentation* und den PEN-Wert der Systemvariablen *PF-KEY.

%RN — Bildschirmdaten-Komprimierung unterdrücken

Anmerkung:

Dieses Kommando ist unter Windows nicht verfügbar.

Kommando-Syntax

%RN

Für den nächsten Bildschirm-I/O unterdrückt dieses Kommando Natural's automatische Bildschirmdaten-Komprimierung und schickt stattdessen den kompletten Schirm.

Normalerweise schickt Natural bei einem Bildschirm-I/O nicht den kompletten Schirm, sondern nur die geänderten Bildschirmdaten. Beim Aufruf eines Nicht-Natural-Programms, das einen Bildschirm-I/O verursacht, erkennt Natural in den meisten TP-Umgebungen, daß ein Nicht-Natural-I/O stattgefunden hat, und schickt beim nächsten Bildschirm-I/O den kompletten Schirm. In manchen TP-Umgebungen erkennt Natural den Nicht-Natural-I/O jedoch nicht; in diesem Fall sollte %RN verwendet werden.

%RN gilt jeweils nur für den nächsten Bildschirm-I/O.

%RO — Bildschirm-Optimierung ein-/ausschalten

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

$$\%RO = \left\{ \begin{array}{c} \text{ON} \\ + \\ \text{T} \\ \text{OFF} \\ - \\ \text{F} \end{array} \right\}$$

Mit diesem Kommando können Sie die automatische Bildschirm-Optimierung von Natural aus- und wieder einschalten.

Die Bildschirm-Optimierung von Natural ermöglicht es, Bildschirm-Daten so komprimiert wie möglich zu senden. Falls dies zu Problemen bei der Bildschirm-Optimierung oder im Zusammenhang mit Hardware-Beschränkungen eines TP-Monitors führen sollte, können Sie die Bildschirm-Optimierung von Natural mit diesem Terminalkommando ausschalten; Bildschirmdaten werden dann in unkomprimierter Form gesendet.

Wenn Sie die Session-Parameter BX=L oder BX=R verwenden, sollten Sie die Bildschirm-Optimierung von Natural ausschalten.

Die folgenden %RO-Kommando-Optionen sind verfügbar:

Kommando	Funktion
%ROON %RO+ %ROT	schaltet die Bildschirm-Optimierung ein.
%ROOFF %RO- %ROF	schaltet die Bildschirm-Optimierung aus.
%RO	schaltet zwischen beiden Modi hin und her.

Anmerkung:

*Anstelle von "ON" können Sie auch "+" oder "T" (true) angeben;
anstelle von "OFF" können Sie auch "-" oder "F" (false) angeben.*

%S — Fortsetzen von NATPAGE

Anmerkung:

Dieses Kommando ist unter Windows nicht verfügbar.

Kommando-Syntax

%S

Mit diesem Terminalkommando können Sie das Aufzeichnen von Bildschirmen durch die Utility NATPAGE wiederaufnehmen (siehe *Natural Utilities Documentation for Mainframes*).

Das Aufzeichnen von Schirmen durch NATPAGE (das mit dem Terminalkommando %P aktiviert wird) kann mit dem Terminalkommando %O unterbrochen werden. Nach einer solchen Unterbrechung können Sie das Aufzeichnen von Schirmen durch NATPAGE mit dem Terminalkommando %S wiederaufnehmen.

Vgl. Terminalkommandos %E, %I, %O und %P.

`%T` — Cursor am Schirmanfang plazieren

Kommando-Syntax

```
%T
```

Dieses Kommando plaziert den Cursor bei der nächsten Bildschirmausgabe in der oberen linken Bildschirmecke.

Anmerkung:

*Auf graphischen Benutzeroberflächen funktioniert dieses Kommando nur, wenn sich in der oberen linken Ecke ein Eingabefeld (AD=A oder AD=M) befindet. Weitere Informationen siehe Kapitel **Session-Parameter**.*

`%Tll/cc` — Cursor in Zeile ll, Spalte cc plazieren

Kommando-Syntax

```
%Tll/cc
```

Dieses Kommando plaziert den Cursor bei der nächsten Bildschirmausgabe in Zeile ll, Spalte cc.

Zeilen und Spalten werden (beginnend mit 1) im physischen Bildschirmfenster gezählt, wobei die Funktionstastenleiste, die Statistikzeile/Infoline (falls aktiv) und die Meldungszeile nicht mitgezählt werden, d.h. “%T1/1” positioniert immer an den Anfang der obersten Datenzeile im Fenster.

Anmerkungen:

- ① *Bei manchen Geräten müssen Sie das Terminalkommando `%T+` ausführen, bevor Sie den Cursor an jede gewünschte Stelle auf dem Bildschirm plazieren können.*
- ② *Auf graphischen Benutzeroberflächen funktioniert `%Tll/cc` nur, wenn sich an der angegebenen Stelle ein Eingabefeld (AD=A oder AD=M) befindet. Weitere Informationen siehe Kapitel **Session-Parameter**.*

%T+/%T- — Cursor in geschützte Felder plazieren

Anmerkungen:

- ① *Diese Kommandos gelten nur für UNIX und OpenVM.*
- ② *Die folgenden Kommandos gelten nur für Siemens-Terminals.*

Kommando-Syntax

`%T {+}`

Die folgenden Kommando-Optionen sind verfügbar:

%T+	Nach Eingabe dieses Kommandos können Sie den Cursor überall auf dem Schirm plazieren, auch in geschützte Felder (die geschützten Felder sind allerdings nach wie vor schreibgeschützt). Dies kann zum Beispiel bei Aktionsleisten-Verarbeitung sinnvoll sein (siehe Terminalkommando %YC).
%T-	Mit diesem Kommando heben Sie die Wirkung von %T+ wieder auf.

`%T*` — Cursor außerhalb des Fensters plazieren

Kommando-Syntax

```
%T*
```

Ist ein Fenster aktiv, das keine Eingabefelder (AD=A oder AD=M) enthält, steht der Cursor normalerweise in der oberen linken Ecke des Fensters (siehe Kapitel **Session-Parameter**).

Mit diesem Terminalkommando können Sie in diesem Fall den Cursor in eine *COM-Systemvariable außerhalb des Fensters plazieren (siehe Kapitel **Systemvariablen**).

Kommando	Funktion
<code>%T*</code>	Schaltet zwischen Cursor-Plazierung in Systemvariable *COM außerhalb des Fensters und Standard-Cursor-Plazierung innerhalb des Fensters hin und her.

`%T*` gilt jeweils nur für das nächste INPUT-Statement und muß *vor* dem betreffenden INPUT-Statement ausgeführt werden.

%T= — Terminalspezifische Converter-Routine

Anmerkung:

Dieses Kommando ist nur relevant, wenn Natural auf Großrechnern unter bestimmten TP-Monitoren verwendet wird.

Kommando-Syntax



Mit diesem Terminalkommando ermöglichen Sie Natural, die für einen bestimmten Terminaltyp geeignete Converter-Routine zu benutzen.

Unter bestimmten TP-Systemen setzt Natural den Terminaltyp automatisch entsprechend der vom TP-System übergebenen Informationen. Falls das TP-System nicht in der Lage ist, Natural über den verwendeten physischen Terminaltyp zu informieren, muß Natural mittels des Terminalkommandos “%T=” mitgeteilt werden, mit welchem Terminaltyp es operiert. Dadurch versetzen Sie Natural in die Lage, die für diesen Terminaltyp erforderliche Attributsequenz aufzubauen.

Standardmäßig (falls keine Informationen vom TP-System übergeben werden) benutzt Natural die Terminaltypen %T=3270 (IBM) bzw. den in PDN definierten Wert, falls dieser nicht durch die Parameter T975X überschrieben wurde (Siemens: vgl. Abschnitt **Installing the Natural TIAM Interface** in der *Natural Installation Documentation for Mainframes*).

Statt des Kommandos %T= können Sie auch den Profilparameter TTYPE verwenden (siehe **Profile Parameters** in der *Natural Parameter Reference Documentation for Mainframes*).

Anmerkung:

Im obigen Diagramm sind nur die gebräuchlichsten Terminaltypen aufgeführt. Falls Sie einen anderen Terminaltyp benutzen, bitten Sie Ihren Natural-Administrator, im NATCONFIG-Modul zu überprüfen, ob dieser Typ mit “%T=” angegeben werden kann.

%TRE — Externe Trace-Funktion aktivieren/deaktivieren

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

$$\%TRE= \left\{ \begin{array}{c} \text{ON} \\ + \\ \text{T} \\ \text{OFF} \\ - \\ \text{F} \end{array} \right\}$$

Dieses Kommando dient dazu, die externe Trace-Funktion zu aktivieren bzw. deaktivieren.

Achtung:

Verwenden Sie dieses Kommando nicht ohne vorherige Rücksprache mit dem Software AG Support. Diese Funktion ist hauptsächlich für Software-AG-interne Verwendung gedacht und dient zur Fehlersuche. Sie schreibt Trace-Daten auf ein externes Trace-Dataset in Abhängigkeit von der TP-Umgebung, in der Natural läuft.

Anmerkungen:

- ① Anstelle von "ON" können Sie auch "+" oder "T" (true) angeben.
Anstelle von "OFF" können Sie auch "-" oder "F" (false) angeben.
- ② Die externe Trace-Funktion kann auch mit dem Profilparameter ETRACE aktiviert / deaktiviert werden. (siehe Abschnitt **Profile Parameters** in der *Natural Parameter Reference Documentation for Mainframes*).

%TRI — Interne Trace-Funktion aktivieren/deaktivieren

Anmerkung:

Dieses Kommando ist nur auf Großrechnern verfügbar.

Kommando-Syntax

$\%TRI = \left\{ \begin{array}{c} ON \\ + \\ T \\ OFF \\ - \\ F \end{array} \right\}$

Dieses Kommando dient dazu, die interne Trace-Funktion zu aktivieren bzw. deaktivieren.

Achtung:

Verwenden Sie dieses Kommando nicht ohne vorherige Rücksprache mit dem Software AG Support. Diese Funktion ist hauptsächlich für Software-AG-interne Verwendung gedacht und dient zur Fehlersuche. Sie übergibt Trace-Daten an die SYSRDC-Utility (die im Abschnitt **Debugging and Monitoring** in der *Natural Utilities Documentation for Mainframes* beschrieben ist).

Anmerkungen:

- ① Anstelle von "ON" können Sie auch "+" oder "T" (true) angeben.
Anstelle von "OFF" können Sie auch "-" oder "F" (false) angeben.
- ② Die interne Trace-Funktion kann auch mit dem Profilparameter ITRACE aktiviert / deaktiviert werden. (siehe Abschnitt **Profile Parameters** in der *Natural Parameter Reference Documentation for Mainframes*).

%U — Umsetzen von Klein- in Großbuchstaben

Kommando-Syntax

```
%U
```

Dieses Kommando bewirkt, daß Natural bei alphanumerischen Eingabedaten automatisch Kleinbuchstaben in Großbuchstaben umsetzt.

Dies ist die Standardeinstellung.

Vgl. Terminalkommando %L (Seite 325).

%V — Steuerung des Print-Modus

Anmerkung:

Dieses Terminalkommando ist nur relevant für Terminals, die invertierten Print-Modus (von rechts nach links) unterstützen.

Kommando-Syntax

%V	<table border="1"> <tr><td>ON</td></tr> <tr><td>OFF</td></tr> <tr><td>N</td></tr> <tr><td>A</td></tr> <tr><td>K</td></tr> </table>	ON	OFF	N	A	K
ON						
OFF						
N						
A						
K						

Die Optionen “N”, “A” und “K” sind nur auf Großrechner-Terminals verfügbar.

%VON	schaltet den invertierten Print-Modus ein, d.h. Daten werden von rechts nach links ausgegeben.
%VOFF	schaltet den normalen Print-Modus ein, d.h. Daten werden von links nach rechts ausgegeben.
%V	schaltet zwischen normalem und invertiertem Print-Modus hin und her.
%VN	schaltet für numerische Felder zwischen normalem und invertiertem Print-Modus hin und her. Dies ist relevant bei sensitiven Terminals, die numerische Werte nicht invertieren.
%VA	bewirkt, daß alphanumerische Felder, die mit PM=I definiert sind, wie numerische Felder behandelt werden. Dadurch ist gewährleistet, daß numerische Werte in solchen Feldern, die andernfalls von rechts nach links interpretiert würden, korrekt von links nach rechts interpretiert werden. Nochmalige Eingabe von %VA schaltet die Wirkung von %VA wieder aus.
%VK	Numerische Felder, für die eine Helproutine oder Editiermaske definiert ist, werden intern von Natural in alphanumerisches Format umgesetzt (sonst könnte ein Benutzer z.B. in ein solches Feld kein Fragezeichen zum Aufrufen der Helproutine eingeben). Bei invertiertem Print-Modus würde diese interne Umsetzung jedoch bewirken, daß ein numerischer Wert in einem solchen Feld von rechts nach links interpretiert würde. Bei numerischen Feldern, die mit PM=I definiert sind und eine Helproutine oder Editiermaske haben, verhindert %VK eine interne Umsetzung in alphanumerisches Format und gewährleistet damit eine korrekte Interpretation der Feldwerte; die korrekte Anwendung einer Editiermaske und die Eingabe eines Fragezeichens ist hierbei nach wie vor gewährleistet. Nochmalige Eingabe von %VK schaltet die Wirkung von %VK wieder aus.

Nähere Informationen zum Print-Modus siehe Session-Parameter PM.

%W — Window-Verarbeitung

Kommando-Syntax

%W

Anmerkung:

Es empfiehlt sich sehr, statt des %W-Kommandos das DEFINE WINDOW-Statement zu verwenden.

Ein Natural-“Window” oder -Fenster ist der Ausschnitt einer von einem Natural-Programm erzeugten logischen Seite, der auf dem Bildschirm zu sehen ist.

Dieses Fenster kann mit dem Terminalkommando %W beeinflußt werden.

Das Kommando muß immer mit Parametern eingegeben werden, um die einzelnen im folgenden beschriebenen Funktionen auszuführen. Sie können mehrere Funktionen gleichzeitig ausführen, indem Sie mehrere Parameter angeben. Zwischen Kommando und Parametern dürfen keine Trennzeichen oder Leerzeichen stehen.

Das Fenster ist ständig vorhanden, auch wenn Sie sich dessen nicht bewußt sind, da die Größe des Fensters — solange Sie sie nicht mit einem DEFINE WINDOW-Statement oder %W-Kommando ändern — mit der Größe Ihres Bildschirms identisch ist. Weitere Informationen über Fensterverarbeitung siehe auch DEFINE WINDOW-Statement in der *Natural Statements-Dokumentation*.

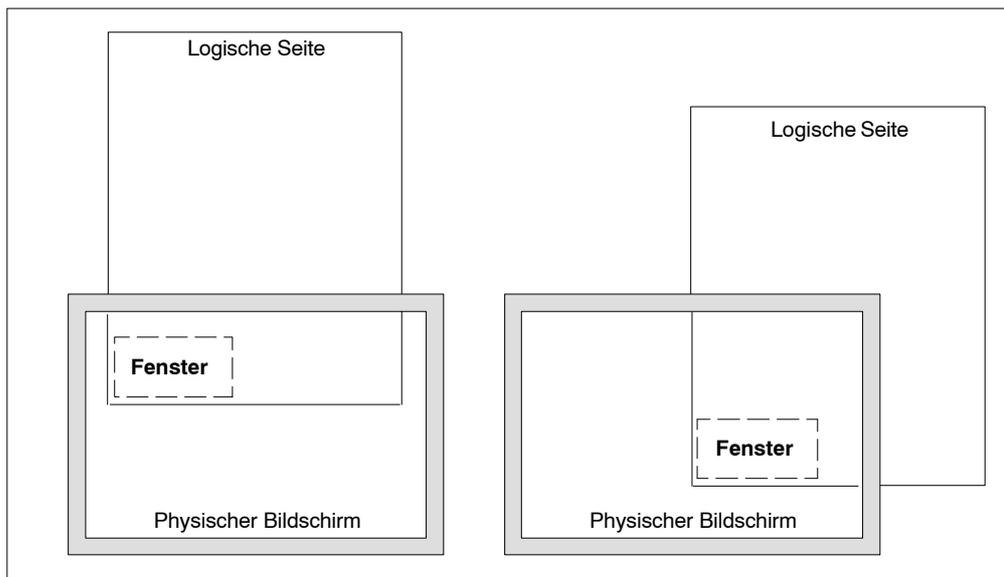
Es gibt zwei Arten von Window-Kommandos:

1. Kommandos, mit denen Sie die Größe und Position des Fensters auf dem *physischen Bildschirm* Ihres Terminals verändern können.
2. Kommandos, mit denen Sie die Position des Fensters auf der vom Programm erzeugten *logischen Seite* verändern können.

1. Größe und Position des Fensters auf dem physischen Bildschirm

Mit den folgenden Window-Kommandos bestimmen Sie die Größe und Position des Fensters auf dem physischen Bildschirm.

Wenn Sie die Position des Fensters auf dem *physischen Bildschirm* ändern, wird die Position des Fensters auf der *logischen Seite* dadurch nicht verändert:



Zu Informationen über mögliche Fenstergrößen siehe DEFINE WINDOW-Statement.

Kommando	Funktion
%WB	Die Fenstergröße (ausschließlich Rahmen) wird auf Bildschirmgröße gesetzt. Falls ein Rahmen definiert ist, ist dieser nicht sichtbar.
%WBlll/ccc	Die obere linke Ecke des Fensters wird in Zeile <i>lll</i> / Spalte <i>ccc</i> plziert (Zeilen und Spalten werden auf dem physischen Bildschirm gezählt). Die Größe des Fensters bleibt dabei gleich. Falls das Fenster zu groß ist, um an die angegebene Stelle plziert zu werden, wird es so nahe wie möglich an diese Stelle plziert.
%WB0	Das Fenster wird an die obere linke Bildschirmecke verschoben. Die Fenstergröße bleibt dabei unverändert.
%W#	Die obere linke Ecke des Fensters wird an die Cursor-Position plziert. Die Größe des Fensters bleibt dabei gleich. Falls das Fenster zu groß ist, um an die angegebene Stelle plziert zu werden, wird es so nahe wie möglich an diese Stelle plziert.
%W?	Die untere rechte Ecke des Fensters wird an die Cursor-Position plziert. Die obere linke Ecke des Fensters verschiebt sich dabei nicht, und die Größe des Fensters wird entsprechend angepaßt.
%WLnn	Die Zeilenlänge (horizontale Ausdehnung) des Fensters (einschließlich Rahmen, falls angegeben) wird auf <i>nn</i> Stellen gesetzt. Wenn Sie <i>nn</i> nicht oder größer als auf den Bildschirm passen würde, angeben, wird die horizontale Ausdehnung auf größtmögliche Zeilenlänge gesetzt, d.h. bis zum rechten Bildschirmrand.
%WCnn	Die Spaltenlänge (vertikale Ausdehnung) des Fensters (einschließlich Rahmen, falls angegeben) wird auf <i>nn</i> Zeilen gesetzt. Wenn Sie <i>nn</i> nicht oder größer als auf den Bildschirm passen würde, angeben, wird die vertikale Ausdehnung auf größtmögliche Spaltenlänge gesetzt, d.h. bis zum unteren Bildschirmrand.

Zeilen- und Spaltenangaben beziehen sich auf die physische Gesamtgröße des Fensters (einschließlich Rahmen, falls angegeben), *nicht* auf die Größe dessen, was innerhalb des Fensters logisch sichtbar ist.

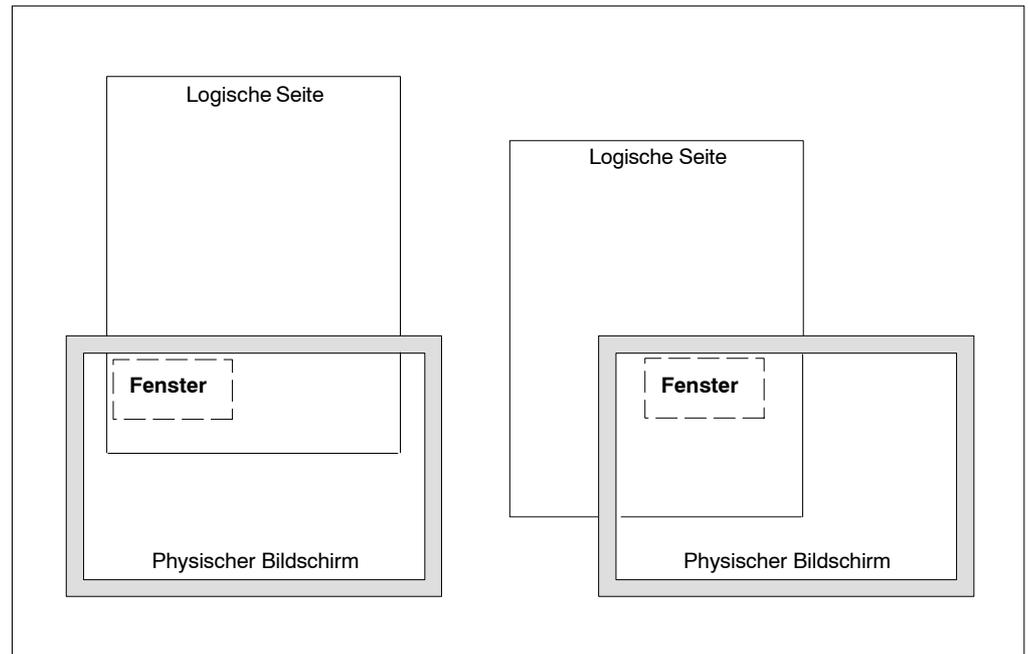
Wird ein Fenster nicht korrekt definiert, so wird ein solches Kommando entweder ignoriert oder die Fenstergröße und -position im Rahmen der physischen Möglichkeiten angepaßt.

Kommando	Funktion
%WF	Schaltet die Rahmung (F = frame) ein. Die Größe des Fensters wird durch einen Rahmen angezeigt. Falls das Fenster kleiner als 4 Zeilen mal 12 (bzw. 13 auf Großrechnern) Spalten ist, ist der Rahmen nicht sichtbar.
%WM	Schaltet die Rahmung aus. Die Größe des Fensters wird nicht durch einen Rahmen angezeigt. Durch Ausschalten des Rahmens ändert sich die Größe des Fensters nicht (lediglich die Größe des im Fenster sichtbaren Seitenausschnitts).
%WO	Die Anzeige von Funktionstastenleiste, Meldungszeile und Statistikzeile wird unterdrückt. Dieses Kommando wirkt nur, falls das betreffende Fenster ein "echtes" Fenster ist (d.h. kleiner als der physische Bildschirm ist). Um die Wirkung von %WO auszuschalten, geben Sie %WO noch einmal ein (oder %WD).
%WP	Standardmäßig werden die Funktionstastenleiste, die Meldungszeile und die Statistikzeile innerhalb eines Fensters angezeigt. Um sie auf dem Schirm außerhalb des Fensters anzuzeigen, verwenden Sie %WP. Um die Wirkung von %WP auszuschalten, verwenden Sie %WD.
%WD	Schaltet die Wirkung von %WF, %WO und %WP (sowie der TITLE-Option des DEFINE WINDOW-Statements) aus.
%WX	Befindet sich außerhalb des Fensters ein *COM-Feld, so ist dieses Feld normalerweise nicht schreibgeschützt. Mit %WX können Sie es schreibgeschützt machen. Diese Option ist nur auf Großrechnern verfügbar.
%WY	Schaltet die Wirkung von %WX wieder aus. Diese Option ist nur auf Großrechnern verfügbar.

2. Position des Fensters auf der logischen Seite

Mit den folgenden Window-Kommandos können Sie die Position des Fensters auf der aktuellen logischen Seite ändern, d.h. der vom Natural-Programm angezeigten Ausgabe oder Map. Die Größe dieser logischen Seite kann über die Größe Ihres Bildschirms hinausgehen.

Wenn Sie die Position des Fensters auf der *logischen Seite* ändern, bleibt die Position des Fensters auf dem *physischen Bildschirm* unverändert. Die logische Seite verschiebt sich also quasi “unter” dem Fenster:



Normalerweise — wenn Sie die Position nicht durch eines der folgenden Kommandos ändern — befindet sich das Fenster an der oberen linken Ecke der logischen Seite, d.h. Sie sehen durch das Fenster den oberen linken Ausschnitt der logischen Seite.

Kommando	Funktion
%W*	Der durch den Cursor auf der Seite markierte Punkt wird an die obere linke Ecke des Fensters geschoben.
%Wll,ccc	Die Seite wird so verschoben, daß sich der durch Zeile <i>lll</i> und Spalte <i>ccc</i> definierte Punkt an der oberen linken Ecke des Fensters befindet. Zeilen und Spalten werden auf der logischen Seite gezählt.
%W<	Die Seite verschiebt sich um eine Fensterbreite nach links.
%W<<	Das Fenster wird bis zum linken Rand der Seite verschoben.
%W<n	Das Fenster wird um <i>n</i> Stellen nach links verschoben ($0 \leq n \leq$ Zeilenlänge der Seite).
%W>	Die Seite verschiebt sich um eine Fensterbreite nach rechts.
%W>>	Das Fenster wird bis zum rechten Rand der Seite verschoben.
%W>n	Das Fenster wird um <i>n</i> Stellen nach rechts verschoben ($0 \leq n \leq$ Zeilenlänge der Seite).
%W+	Die Seite verschiebt sich um eine Fensterhöhe nach unten. (*)
%W++	Das Fenster wird ans untere Ende der Seite verschoben. (*)
%W+n	Das Fenster wird um <i>n</i> Zeilen nach unten verschoben ($0 \leq n \leq$ Spaltenlänge der Seite). (*)
%W-	Die Seite verschiebt sich um eine Fensterhöhe nach oben.
%W-	Das Fenster wird ans obere Ende der Seite verschoben.
%W-n	Das Fenster wird um <i>n</i> Zeilen nach oben verschoben ($0 \leq n \leq$ Spaltenlänge der Seite).
%WH	Standardmäßig wird die Fenster-Position auf der logischen Seite bei einem Bildschirm-I/O wieder auf "obere linke Ecke" zurückgesetzt. %WH bewirkt, daß die gesetzte Fenster-Position beim nächsten I/O nicht zurückgesetzt wird, sondern erhalten bleibt. %WH gilt jeweils nur für den nächsten I/O.
%WS	Die STAY-Option ist eingeschaltet; d.h. die Kontrolle bleibt (englisch: "stay") auf der aktuellen Seite, bis das Ende der Seite erreicht ist. Wenn eine Seite in vertikaler Richtung noch nicht vollständig angezeigt worden ist, wird Ihnen dies durch die Meldung "VVVV" angezeigt. Jedesmal, wenn Sie EINGABE drücken, verschiebt sich die Seite um ein Fenster nach unten, bis das Seitenende erreicht ist. Mit der nächsten EINGABE wird dann die Kontrolle wieder an das Programm übergeben. (Dies gilt nicht für Seiten, die über ein INPUT-Statement mit Eingabefeldern (AD=A oder AD=M) erzeugt wurden.)
%WN	Die STAY-Option ist ausgeschaltet. Wenn Sie EINGABE drücken, wird die Kontrolle an das Programm übergeben.

* Das Fenster kann höchstens bis zur letzten nicht leeren Zeile der Seite verschoben werden.

Anmerkungen:

- ① *Wenn Sie eines der obigen Kommandos zum Verschieben des Fensters in einem Programm verwenden wollen, weisen Sie das Kommando einer Funktionstaste zu (mit einem SET KEY-Statement).*
- ② *Wenn Sie es mit einem SET CONTROL-Statement angeben wollen, muß diesem ein REINPUT-Statement folgen (d.h. es muß zwischen dem REINPUT-Statement und dem entsprechenden INPUT-Statement stehen); sonst kann Natural das Kommando nicht eindeutig einem bestimmten Fenster zuordnen (und ignoriert es).*
- ③ *Allerdings sollte zwischen einem INPUT-Statement mit Option WINDOW='window-name' und dem dazugehörigen REINPUT-Statement grundsätzlich kein SET CONTROL 'W'-Statement stehen.*

Beispiele für Kommando-Kombinationen

Sie können die verschiedenen Parameter des %W-Kommandos auch miteinander kombinieren. Zum Beispiel:

%W<<—	Das Fenster wird an die obere linke Ecke der Seite geschoben.
%W>>++	Das Fenster wird an die untere rechte Ecke der Seite geschoben.
%W+—	Der (in vertikaler Richtung) vorletzte Fensterausschnitt der Seite wird gezeigt.
%W+3>6	Das Fenster verschiebt sich auf der Seite um 3 Zeilen nach unten und 6 Spalten nach rechts.
%W10+>	Das Fenster wird an Zeile 10 der Seite geschoben, dann um ein Fenster nach unten und um ein Fenster nach rechts.
%WL40C10++—3	Das Fenster wird mit einer Zeilenlänge von 40 Stellen und einer Spaltenlänge von 10 Zeilen definiert, es wird ans untere Ende der logischen Seite und dann wieder 3 Zeilen nach oben geschoben.
%WL30C10B3/15—<<	Das Fenster wird mit einer Zeilenlänge von 30 Stellen und einer Spaltenlänge von 10 Zeilen definiert, es wird in Zeile 3 und Spalte 15 des physischen Bildschirms plaziert und an die obere linke Ecke der logischen Seite geschoben.
%WFS	Um das Fenster wird ein Rahmen generiert, und die STAY-Option wird eingeschaltet.

Anmerkungen:

- ① Wenn Sie mehrere Parameter mit dem %W-Kommando angeben, beachten Sie bitte, daß Sie auf Großrechnern nach dem “%” höchstens 24 Zeichen angeben können; alle weiteren Zeichen werden ignoriert.
- ② Die Parameter werden der Reihe nach ausgewertet, so daß unterschiedliche Reihenfolgen derselben Parameter zu unterschiedlichen Ergebnissen führen können.

%WA und %WZ — Bildschirm speichern vor Fenster

Kommando	Funktion
%WA	<p>Mit “%WA” aktivieren Sie die Funktion “Bildschirm speichern vor Fenster”. Wenn diese Funktion aktiv ist und ein Fenster geöffnet wird, werden alle aktiven Bildschirmdaten, die von dem Fenster überlagert werden, vorher gespeichert. Wenn das Fenster verschoben wird, wird der gespeicherte Bildschirm rekonstruiert, bevor das Fenster an einer anderen Stelle auf dem Schirm aufgebaut wird. Es ist auch möglich, mehrere aufgerufene Fenster zu rekonstruieren, wenn das aufrufende Fenster wieder aktiv wird.</p> <p>Wenn das aktuelle INPUT-Statement ein Fenster verwendet, wird der Bildschirm gespeichert, bevor das Fenster ausgegeben wird. Wenn dasselbe INPUT-Statement wiederholt wird, wird der aktuelle Bildschirm bzw. alle anschließend gespeicherten Bildschirme rekonstruiert und wieder auf den Schirm zurückgeschrieben. (Auf Großrechnern werden die Bildschirme im WSIZE-Puffer gespeichert, dessen Größe mit dem Profilparameter WSIZE bestimmt werden kann; siehe Profile Parameters in der <i>Natural Parameter Reference Documentation</i>.)</p> <p>Diese Funktion ermöglicht beispielsweise die Verwendung von Fenstern in PC-ähnlicher Form. Für ein bestimmtes Fenster können beliebig viele aufgerufene, “abhängige” Fenster gespeichert werden. All diese Fenster verschwinden vom Schirm, wenn das Haupteingabefenster wiederausgeführt wird.</p> <p>Der Inhalt des Puffers (d.h. die gespeicherten Schirme) wird gelöscht, wenn Natural wieder einen ganzen Bildschirm ausgibt, wenn Natural in den Kommandomodus (NEXT) gelangt, nach einem LOGON-Kommando oder nach Drücken der CLEAR- bzw. LÖSCH-Taste.</p>
%WZ	Mit “%WZ” deaktivieren Sie ein vorhergegangenes %WA-Kommando.

%X — Steuerung der Statistikzeile/Infoline

Kommando-Syntax

$$\%X \left[\begin{array}{c} \left[\begin{array}{c} I \\ S \end{array} \right] \left[\begin{array}{c} + \\ - \end{array} \right] \\ B \\ T \\ nn \\ -nn \end{array} \right]$$

Dieses Terminalkommando steuert die Anzeige der Natural-Statistikzeile/Infoline.

Natural für Großrechner:

Auf Großrechnern kann die Zeile als Statistikzeile oder als Infoline benutzt werden (aber nicht beides gleichzeitig); auf allen anderen Plattformen kann sie nur als Infoline benutzt werden.). Siehe auch **Statistics Line/Infoline – Terminal Command %X** im Abschnitt **Designing User Interfaces**, Unterabschnitt **Screen Design** im *Natural User's Guide for Mainframes*.

Natural für UNIX/OpenVMS, Natural für Windows:

Die Zeile kann nur als Infoline benutzt werden.

Kommando	Funktion
% X+	Schaltet die Anzeige der Statistikzeile/Infoline ein.
% X-	Schaltet die Anzeige der Statistikzeile/Infoline aus.
% X	Schaltet die Anzeige der Statistikzeile/Infoline ein bzw. wieder aus.
% XI+	Zeigt die Zeile als Infoline an.
% XI-	Zeigt die Zeile als Statistikzeile an (nur auf Großrechnern möglich).
% XI	Schaltet zwischen Infoline- und Statistik-Anzeige hin und her (nur auf Großrechnern möglich).
% XS+	Zeigt zusätzliche Statistik-Informationen an.
% XS-	Zeigt wieder die ursprünglichen Statistik-Informationen an.
% XS	Schaltet zwischen zusätzlichen und ursprünglichen Statistik-Informationen hin und her.
% XB	Zeigt die Statistikzeile/Infoline am unteren Bildschirmrand an (B = Bottom).
% XT	Zeigt die Statistikzeile/Infoline am oberen Bildschirmrand an (T = Top).
% Xnn	Zeigt die Statistikzeile/Infoline in der nn -ten Zeile des Schirms an. Ist die angegebene Zeilennummer nn außerhalb des aktuellen Schirms, wird die Statistikzeile/Infoline nicht angezeigt.
% X$-nn$	Zeigt die Statistikzeile/Infoline in der nn -ten Zeile von unten auf dem Schirm an. Ist die angegebene Zeilennummer $-nn$ außerhalb des aktuellen Schirms, wird die Statistikzeile/Infoline nicht angezeigt.

Infoline

Daten können in die Infoline geschrieben werden, indem Sie "INFOLINE" als Ausgabeziel im DEFINE PRINTER-Statement angeben. Es kann nur eine einzelne Zeile in die Infoline geschrieben werden. Die Infoline kann zur Anzeige von Status-Informationen verwendet werden, zum Beispiel bei der Fehlersuche; sie kann auch als Trennlinie (Separator Line; wie in den SAA-Standards definiert) verwendet werden.

Statistikzeile (nur auf Großrechnern)

Wenn die Zeile als Statistikzeile verwendet wird, liefert sie folgende Informationen:

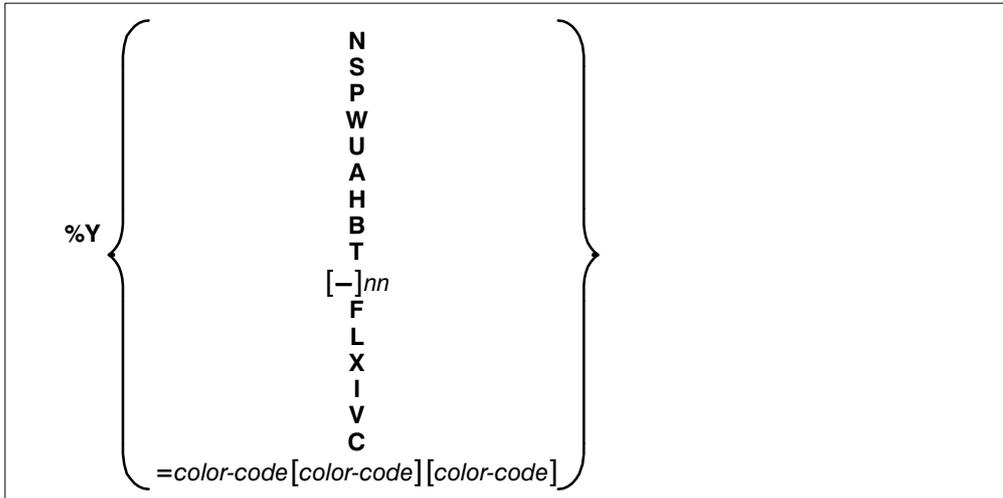
Kommando-Option	Funktion
IO	Anzahl der während der letzten Bildschirmoperation an den Bildschirm übergebenen Bytes.
AIO	Durchschnittliche Anzahl der je Bildschirmoperation an den Bildschirm übergebenen Bytes (seit Beginn der Natural-Session).
L	Logische Zeilennummer (L = Line) der oberen Zeile des aktiven Natural-Windows, gezählt auf der logischen Seite.
C	Logische Spaltennummer (C = Column) der äußersten linken Zeile des aktiven Natural-Windows, gezählt auf der logischen Seite.
LS	Logische Zeilenlänge (LS = Line Size) der aktuellen logischen Seite (wie mit dem Session-Parameter LS definiert).
PS	Logische Seitenlänge (PS = Page Size) der aktuellen logischen Seite (wie mit dem Session-Parameter PS definiert).
PLS	Physische Zeilenlänge des aktiven Natural-Fensters.
PCS	Physische Spaltenlänge des aktiven Natural-Fensters.
FLD	Die Anzahl der auf dem letzten Schirm generierten Felder.
CLS	Die Anzahl der Programm-Aufrufe während des letzten Terminal-I/Os.
ADA	Die Anzahl der Adabas-Aufrufe während des letzten Terminal-I/Os.

Zusätzliche Informationen, die mit %XS+ angezeigt werden:

Kommando-Option	Funktion
MIN	Kürzeste Dauer (in Sekunden) der Natural-Aktivitäten zwischen zwei Terminal-I/Os während der laufenden Natural-Session.
MAX	Längste Dauer (in Sekunden) der Natural-Aktivitäten zwischen zwei Terminal-I/Os während der laufenden Natural-Session.
AVR	Durchschnittliche Dauer (in Sekunden) der Natural-Aktivitäten zwischen zwei Terminal-I/Os während der laufenden Natural-Session.
LST	Dauer (in Sekunden) der Natural-Aktivitäten zwischen dem vorletzten und dem vorvorletzten Terminal-I/O.

%Y — Steuerung der PF-Tastenleiste

Kommando-Syntax



Das Terminalkommando %Y steuert die Anzeige der Natural-Funktionstastenleiste.

Anmerkung:

Auf graphischen Benutzeroberflächen wird dieses Kommando ignoriert.

Anzeigeformat der Funktionstastenleiste:

Kommando	Funktion
%YN	Zeigt die Funktionstastenleiste im normalen tabularischen Software-AG-Format an.
%YS	Zeigt die Funktionstastenleiste im sequentiellen Format an, d.h. die Tasten werden nacheinander angezeigt, und zwar nur die Tasten, denen Namen zugewiesen wurden (PF1=wert,PF2=wert usw.).
%YP	Zeigt die Funktionstastenleiste im sequentiellen PC-Format; entspricht %YS, außer daß vor den Namen "Fn=" statt "PFn=" steht.
%YW	Entspricht %YP — gilt jedoch nur, wenn die Funktionstastenleiste innerhalb eines Fensters angezeigt wird. Diese Option ist nur auf Großrechnern verfügbar.
%YU	Hebt die Wirkung von %YW wieder auf. Diese Option ist nur auf Großrechnern verfügbar.

Ein- oder zweizeilige Anzeige:

Kommando	Funktion
%YA	(alle) Zeigt beide PF-Tastenzeilen an.
%YH	(halb) Zeigt nur eine PF-Tastenzeile an: beim normalen Software-AG-Anzeigeformat (%YN) ist dies die Zeile mit den Funktionstasten-Namen; bei den anderen beiden Anzeigeformaten (%YS und %YP) ist dies die obere Zeile.

Positionierung der Funktionstastenleiste:

Kommando	Funktion
%YB	Zeigt die Funktionstastenleiste am unteren Bildschirmrand an (B = Bottom).
%YT	Zeigt die Funktionstastenleiste am oberen Bildschirmrand an (T = Top).
%Ynn	Zeigt die Funktionstastenleiste in der <i>nn</i> -ten Zeile des Schirms an.
%Y-nn	Zeigt die Funktionstastenleiste in der <i>nn</i> -ten Zeile von unten auf dem Schirm an. Ist die angegebene Zeilennummer <i>nn</i> oder <i>-nn</i> außerhalb des aktuellen Schirms, wird die Funktionstastenleiste nicht angezeigt.

Bereich der angezeigten Funktionstasten:

Kommando	Funktion
%YF	Zeigt den ersten Bereich von Funktionstasten (d.h. in der Regel 1 bis 12) an (F = First).
%YL	Zeigt den letzten Bereich von Funktionstasten (d.h. in der Regel 13 bis 24) an (L = Last).
%YX	Schaltet zwischen beiden Anzeigen hin und her.

Intensivierte bzw. inverse Anzeige der Funktionstastenleiste:

Kommando	Funktion
%YI	Zeigt die Funktionstastenleiste intensiviert an. Durch nochmaliges Eingeben von %YI erhalten Sie wieder die normale nicht intensivierte Anzeige.
%YV	Zeigt die Funktionstastenleiste invers an. Durch nochmaliges Eingeben von %YV erhalten Sie wieder die normale nicht inverse Anzeige.

Farbe der Funktionstastenleiste:

Kommando	Funktion
<code>%Y=color-code(s)</code>	<p>Zeigt die Funktionstastenleiste in den angegebenen Farben an. Sie können die gleichen Farbcodes angeben wie beim Session-Parameter CD.</p> <p>Sie können bis zu drei <i>color-codes</i> angeben: Der erste <i>color-code</i> gilt für die erste Funktionstastenzeile (in der die Funktionstastennummern angezeigt werden), der zweite <i>color-code</i> gilt für die zweite Funktionstastenzeile (in der die Namen, die den Funktionstasten zugeordnet sind, angezeigt werden), und der dritte <i>color-code</i> gilt für den Hintergrund beider Zeilen.</p> <p>Mit <code>%Y=GRPIYE</code> würden Sie beispielsweise die Zeichen der ersten Zeile in Grün, die Zeichen der zweiten Zeile in Pink, und den Hintergrund beider Zeilen in Gelb erhalten.</p> <p><i>Anmerkung: Auf Großrechnern wird der dritte "colour-code" ignoriert.</i></p>

Cursor-Sensitivität:

Kommando	Funktion
<code>%YC</code>	<p>Dieses Kommando macht die Funktionstastenleiste cursor-sensitiv. Sie reagiert dann wie eine Aktionsleiste auf einem PC-Bildschirm: der Benutzer wählt mit dem Cursor lediglich den Namen oder die Nummer der gewünschten Funktionstaste aus und drückt EINGABE, und Natural reagiert, als ob die betreffende Funktionstaste gedrückt worden wäre.</p> <p>Durch nochmaliges Eingeben von <code>%YC</code> schalten Sie die Cursor-Sensitivität wieder aus.</p> <p>Durch Verwendung von <code>%YC</code> in Verbindung mit Software-AG-Anzeigeformat (<code>%YN</code>) und einzeliger Anzeige (<code>%YH</code>) können Sie Ihre Anwendungen mit einer sehr komfortablen Aktionsleisten-Verarbeitung ausstatten: der Benutzer wählt dann bloß noch den Namen einer Funktion mit dem Cursor aus und drückt EINGABE, und die Funktion wird ausgeführt.</p>

Um die Anzeige der Funktionstastenleiste in einem Programm zu aktivieren, verwenden Sie den Session-Parameter `KD=ON` (siehe Kapitel **Session-Parameter**).

`%Z` — Arbeitsbereich des Editors löschen

Kommando-Syntax

```
%Z
```

Dieses Kommando löscht den Inhalt des Editor-Arbeitsbereichs.

Anmerkung:

Dieses Kommando kann nur mit einem SET CONTROL-Statement verwendet werden.

SCHLÜSSELWÖRTER UND RESERVIERTE WÖRTER

Dieses Kapitel enthält eine Liste aller Natural-Schlüsselwörter und für Natural reservierten Wörter.

Um mögliche Namenskonflikte zu vermeiden, empfiehlt es sich sehr, diese Schlüsselwörter oder reservierten Wörter oder Bestandteile derselben nicht als Namen für Ihre Daten oder Prozeduren zu benutzen.

Die Statement-Schlüsselwörter, auf die der Natural-Profilparameter KC (Keyword Checking) bzw. die KCHECK-Option des NTCMPO-Makros oder das COMPOPT-Systemkommando (vgl. *Natural Referenz-Dokumentation*) wirken, sind **fett und kursiv** geschrieben.

.	*DATN	*OCCURRENCE
<	*DATU	*OI
<>	*DATX	*OPSYS
<=	*DEVICE	*OS
+	*DIALOG-ID	*OSVERS
+V	*ERROR	*OUT
*	*ERROR-LINE	*OUTIN
**	*ERROR-NR	*PAGE-NUMBER
*CONVID	*ERROR-TA	*PAGESIZE
*COUNTER	*EVENT	*PF-KEY
*CURS-COL	*HARDCOPY	*PROGRAM
*CURS-LINE	*HARDWARE	*ROWCOUNT
*CURSOR	*IN	*TIMD
*DAT4D	*INIT-ID	*TIME
*DAT4E	*INIT-PROGRAM	*TIMESTMP
*DAT4I	*INIT-USER	*TIMN
*DAT4J	*ISN	*TIMX
*DAT4U	*LIBRARY-ID	*TPSYS
*DATD	*LINE-COUNT	*UI
*DATE	*LINESIZE	*WINDOW-LS
*DATG	*MACHINE-CLASS	*WINDOW-POS
*DATI	*NUMBER	*WINDOW-PS
*DATJ	*OCC	*WINMGR

*WINMGRVERS	ASC	CAT
^<	ASCENDING	CATALL
^>	<i>ASSIGN</i>	CATALOG
^=	ASSIGNING	CATLG
-	AT	CC
/	<i>AT BREAK</i>	CD
>	<i>AT END</i>	CDID
>=	<i>AT START</i>	CF
?	<i>AT TOP</i>	CHAR
:=	<i>ATN</i>	CHECK
=	AUTO	CHILD
A	<i>AVER</i>	CIPH
<i>A-AVER</i>	AVG	CIPHER
<i>A-MAX</i>	<i>BACKOUT</i>	CLASS
<i>A-MIN</i>	BASE	CLEAR
<i>A-NAVER</i>	BATCH	<i>CLOSE CONVERSATION</i>
<i>A-NCOUNT</i>	<i>BEFORE</i>	<i>CLOSE LOOP</i>
<i>A-NMIN</i>	BETWEEN	<i>CLOSE PC</i>
<i>A-SUM</i>	<i>BLOCK</i>	<i>CLOSE PRINTER</i>
<i>ABS</i>	BLOCKE	<i>CLOSE WORK</i>
ABSOLUTE	BLOCKED	CLR
<i>ACCEPT</i>	BOT	CMS
ACTION	BOTTOM	COM
AD	<i>BREAK</i>	COMMAND
<i>ADD</i>	<i>BROWSE</i>	<i>COMMIT</i>
ADHOC	BUT	<i>COMPOSE</i>
AFTER	BUT NOT	<i>COMPRESS</i>
AL	BX	<i>COMPUTE</i>
ALARM	BY	CONDITION
<i>ALL</i>	C	CONST
AND	CABINET	CONSTANT
AND TRANSLATE	<i>CALL</i>	CONTEXT
<i>ANY</i>	CALLING	CONTROL
APPLIC-ID	<i>CALLNAT</i>	CONVERSATION
APPLIC-NAME	CAP	COPIES
AS	CAPTIONED	<i>COPY</i>

<i>COS</i>	DISTINCT	<i>END-FILE</i>
<i>COUNT</i>	<i>DIVIDE</i>	<i>END-FIND</i>
COUPLED	<i>DLOGOFF</i>	<i>END-FOR</i>
CR	<i>DLOGON</i>	<i>END-HISTOGRAM</i>
CREATE	<i>DNATIVE</i>	<i>END-IF</i>
CURRENT	<i>DO</i>	END-INTERFACE
CURS-FIELD	DOCUMENT	<i>END-JOIN</i>
CURSOR	<i>DOEND</i>	<i>END-LOOP</i>
CV	<i>DOWNLOAD</i>	END-METHOD
DATA	<i>DRAW</i>	<i>END-NOREC</i>
DATAAREA	DU	END-PARAMETERS
DATE	DUMP	<i>END-PROCESS</i>
DAY	DY	END-PROPERTY
DAYS	DYNAMIC	<i>END-READ</i>
DC	E	<i>END-REPEAT</i>
DEBUG	EDIT	<i>END-SELECT</i>
<i>DECIDE</i>	EDITED	END-SERVER
DECIMAL	EDT	<i>END-SORT</i>
<i>DEFINE</i>	EJ	<i>END-START</i>
<i>DEFINE DATA</i>	<i>EJECT</i>	<i>END-SUBROUTINE</i>
<i>DEFINE SERVER</i>	<i>ELSE</i>	<i>END-TOPPAGE</i>
<i>DEFINE VIEW</i>	EM	<i>END-UNITE</i>
<i>DELETE</i>	<i>END</i>	END-VALUE
DELIMITED	<i>END-ACTION</i>	END-VALUES
DELIMITER	<i>END-ALL</i>	END-VIEW
DELIMITERS	<i>END-BEFORE</i>	<i>END-WORK</i>
DESC	<i>END-BLOCK</i>	<i>ENDHOC</i>
DESCENDING	<i>END-BREAK</i>	ENDING
DEST	<i>END-BROWSE</i>	ENDING AT
DESTINATION	END-CLASS	ENTER
DIALOG	<i>END-DECIDE</i>	<i>ENTIRE</i>
DIALOG-ID	END-DEFINE	ENTR
DIGITS	<i>END-ENDDATA</i>	EQ
DISP	<i>END-ENDFILE</i>	EQUAL TO
<i>DISPLAY</i>	<i>END-ENDPAGE</i>	ERROR
<i>DISPLAY FORMATTED</i>	<i>END-ERROR</i>	ERROR-LINE

ERROR-TA	FORMATTED	HEX
ERRORS	FORMATTING	HISTOGRAM
ES	FORMS	HORIZ
ESCAPE	FOUND	HORIZONTALLY
ETID	FRAC	HOURL
EVEN	FRAMED	HW
EVENT	FROM	IA
EVERY	FS	IC
EX	FULL	ID
EXAMINE	FUNCTIONS	IDENTICAL
EXEC	G	IF
EXECUTE	GC	IGNORE
EXISTS	GDA	IM
EXIT	GE	IMMEDIATE
EXP	GEN	IMPORT
EXPORT	GENERATED	IN
EXTRACTING	GET	INCCONT
F	GFID	INCDIC
FALSE	GIVE	INCDIR
FC	GIVING	INCLUDE
FETCH	GLOBAL	INCLUDED
FIELD	GLOBALS	INCMAC
FIELDS	GRAPHICS	INDEPENDENT
FILE	GREATER EQUAL	INDEX
FILES	GREATER THAN	INDEXED
FILL	GROUP	INDICATOR
FILLER	GROUP BY	INDX
FIN	GT	INIT
FINAL	H	INITIAL
FIND	HAVING	INPL
FIRST	HC	INPUT
FL	HD	INSERT INTO
FLOAT	HE	INT
FOR	HEADER	INTEGER
FORM	HELLO	INTERCEPTED
FORMAT	HELP	INTERFACE

INTERMEDIATE	LOG-LS	MT
INTO	LOG-PS	MULTIPLY
INVERTED	LOGICAL	NAME
INVESTIGATE	LOGOFF	NAMED
IO	LOGON	NAVER
IP	LOOP	NC
IS	LOWER	NCOUNT
ISN	LOWER CASE	NE
ISPF	LS	NET
JOIN	LT	NEWPAGE
JUST	M	NL
JUSTIFIED	MACROAREA	NMIN
KD	MAIL	NO
KEY	MAINMENU	NO ERASE
KEYS	MAP	NO PARAMETER
L	MARK	NO PARMS
LANGUAGE	MASK	NODE
LAST	MAX	NOHDR
LC	MC	NONE
LE	MCG	NOT
LEAVE	MESSAGES	NOT <
LEAVING	METHOD	NOT >
LEFT	MGID	NOT =
LENGTH	MICRO	NOT EQ
LESS	MICROSECOND	NOT GT
LESS EQUAL	MIN	NOT LT
LESS THAN	MINUTE	NOT EQUAL
LEVEL	MIX	NOTIT
LIKE	MODIFIED	NOTITLE
LIMIT	MODULES	NPC
LINDICATOR	MONTH	NPLCMD1
LINES	MORE	NPLCMD2
LIST	MOVE	NPLCMD3
LISTED	MOVING	NULL
LOCAL	MP	NULL-HANDLE
LOG	MS	NUMBER

NUMERIC	PEN	REMAINDER
O	PERFORM	RENAME
OBJECT	PF-NAME	RENUM
OBTAIN	PF n ($n = 1$ bis 9)	RENUMBER
OF	PF nn ($nn = 10$ bis 99)	REPEAT
OFF	PGM	REPEATED
OFFSET	PHYSICAL	REPLACE
OLD	PLOT	REPORT
ON	PM	REQUEST
ON ACTION	POS	REQUIRED
ON ERROR	POSITION	RESET
ONCE	PR	RESETTING
OPEN	PREV	RESPONSE
OPEN CONVERSATION	PREVIOUS	RESTORE
OPTIONS	PRIMARY	RESULT
OR	PRINT	RET
OR =	PRINTER	RETAIN
OR EQ	PRIORITY	RETAINED
OR EQUAL	PROCESS	RETRY
OR EQUAL TO	PROCESSING	RETURN
OR=	PROFILE	REVERSED
ORDER BY	PROGRAM	RIGHT
OUTPUT	PROGRAMS	ROLLBACK
OVFLW	PROPERTY	ROUNDED
PAGE	PS	ROUTINE
PARAMETER	PURGE	RULEVAR
PARAMETERS	QUARTER	RUN
PARENT	R	RUNMODE
PASSW	READ	SA
PASSWORD	READONLY	SAME
PATTERN	RECORD	SAVE
PA1	RECORDS	SCAN
PA2	REDEFINE	SCR
PA3	REINPUT	SCRATCH
PC	REJECT	SCREEN
PD	RELEASE	SEARCH

SECOND	STARTUP	TIMESTAMP
SELECT	STATEMENT	TIMEZONE
SELECTION	STATUS	TITLE
SEND	STEP	TO
SEPARATE	STEPLIB	TO VARIABLE
SEQUENCE	STOP	TO VARIABLES
SERVER	STORE	TOP
SET	STOW	TOTAL
SET TIME	SUBPROGRAM	TP
SETS	SUBPROGRAMS	TR
SETTIME	SUBROUTINE	TRAILER
SETUP	SUBSTR	TRANSACTION
SF	SUBSTRING	TRANSFER
SG	SUBTRACT	TRANSLATE
SGN	SUM	TREQ
SHORT	SUPPRESS	TRUE
SHOW	SUPPRESSED	TS
SIN	SUSPEND	TSO
SINGLE	SYMBOL	TYPE
SIZE	SYSTEM	U
SKIP	T	UC
SL	TAN	UNCAT
SM	TC	UNCATALOG
SOME	TECH	UNCATLG
SORT	TERMINATE	UNDERLINED
SORTED	TEST	UNDLIN
SOUND	TEXT	UNION
SOURCE	TEXTAREA	UNIQUE
SPACE	TEXTVARIABLE	UNITE
SPECIFIED	THAN	UNTIL
SQLID	THEM	UPDATE
SQRT	THEN	UPLOAD
STACK	THRU	UPPER
START	TIME	UPPER CASE
STARTING	TIME-OUT	USED
STARTING FROM	TIMES	USER

USER-NAME	VERT	WITH
USING	VERTICALLY	WORK
<i>VAL</i>	VIA	<i>WRITE</i>
<i>VALUE</i>	VIEW	X
<i>VALUES</i>	<i>WASTE</i> PAPER	XREF
VARGRAPHIC	WH	YEAR
VARIABLE	<i>WHEN</i>	ZD
VARIABLES	WHERE	ZP
VERIFY	<i>WHILE</i>	
VERSIONS	WINDOW	

INDEX

Symbole

- *APPLIC-ID-Systemvariable, 187
- *APPLIC-NAME-Systemvariable, 187
- *COM-Systemvariable, 188
 - Daten kopieren nach, 308
- *COUNTER-Systemvariable, 190
- *CPU-TIME-Systemvariable, 190
- *CURS-COL-Systemvariable, 191
- *CURS-FIELD-Systemvariable, 191, 275
- *CURS-LINE-Systemvariable, 192
- *CURSOR-Systemvariable, 193
- *DATA-Systemvariable, 193
- *DEVICE-Systemvariable, 194
- *ERROR-LINE-Systemvariable, 195
- *ERROR-NR-Systemvariable, 195
- *ERROR-TA-Systemvariable, 196
- *ETID-Systemvariable, 196
- *GROUP-Systemvariable, 197
- *HARDCOPY-Systemvariable, 198
- *HARDWARE-Systemvariable, 198
- *HOSTNAME-Systemvariable, 198
- *INIT-ID-Systemvariable, 199
- *INIT-PROGRAM-Systemvariable, 200
- *INIT-USER-Systemvariable, 200
- *ISN-Systemvariable, 202
- *LANGUAGE-Systemvariable, 203
- *LENGTH-Systemvariable, 206
- *LEVEL-Systemvariable, 206
- *LIBRARY-ID-Systemvariable, 206
- *LINE-COUNT-Systemvariable, 207
- *LINESIZE-Systemvariable, 207
- *LOG-LS-Systemvariable, 208
- *LOG-PS-Systemvariable, 208
- *MACHINE-CLASS-Systemvariable, 208
- *NATVERS-Systemvariable, 209
- *NET-USER-Systemvariable, 209

Natural Referenzhandbuch

- *NUMBER-Systemvariable, 210
- *OCCURRENCE-Systemvariable, 211
- *OPSYS-Systemvariable, 213
- *OS-Systemvariable, 214
- *OSVERS-Systemvariable, 214
- *PAGE-NUMBER-Systemvariable, 215
- *PAGESIZE-Systemvariable, 215
- *PARM-USER-Systemvariable, 216
- *PATCH-LEVEL-Systemvariable, 216
- *PF-KEY-Systemvariable, 217
- *PF-NAME-Systemvariable, 218
- *PID-Systemvariable, 218
- *PROGRAM-Systemvariable, 218
- *ROWCOUNT-Systemvariable, 219
- *SCREEN-IO-Systemvariable, 219
- *SERVER-TYPE-Systemvariable, 220
- *STARTUP-Systemvariable, 221
- *STEPLIB-Systemvariable, 222
- *SUBROUTINE-Systemvariable, 223
- *THIS-OBJECT-Systemvariable, 223
- *TPSYS-Systemvariable, 224
- *UI-Systemvariable, 224
- *USER-NAME-Systemvariable, 225
- *USER-Systemvariable, 225
- *WINDOW-LS-Systemvariable, 225
- *WINDOW-POS-Systemvariable, 226
- *WINDOW-PS-Systemvariable, 226
- *WINMGR-Systemvariable, 227
- *WINMGRVERS-Systemvariable, 227

A

Abbrechen

Natural-Operation, Terminalkommandos *%.* und *%%*, 293

Verarbeitungsschleife, RESET/ENTER-Tasten, 291

Abend. *Siehe* Abbrechen

Abrunden. *Siehe* Rundungsregeln

ABS, mathematische Funktion, 270

Absoluter Wert eines Feldes, ABS-Funktion, 270

Adabas

Interne Satznummer. *Siehe* ISN

Transaktionskennung. *Siehe* ETID

- Addition. *Siehe* Arithmetik
- AD-Parameter, 104
- Aktionsleiste, 367
- Alarmton, SA-Parameter, 175
- AL-Parameter, 108
- Alphanumerisches Feld
 - Ausgabelänge, AL-Parameter, 108
 - Editiermaske für, 131
 - Format A, 11
 - Konstante, 32
 - numerischer Wert in, VAL-Funktion, 270
- Alter Feldwert, OLD-Systemfunktion, 264
- Anwendung. *Siehe* Library
- Anzeigen, von Eingabedaten, unterdrücken, Terminalkommando *%**, 296
- Applikation. *Siehe* Library
- Arcustangens eines Feldes, ATN-Funktion, 270
- Arithmetik, 75
 - arithmetischer Ausdruck in logischer Bedingung, 48
 - Ergebnisfelder, 80
 - Ergebnisgenauigkeit, 87
 - Fehlerbedingungen, 89
 - mit Datum, 83
 - mit Gleitkomma, 81
 - mit Zeit, 83
 - Performance-Hinweise, 86
- Array
 - Siehe auch* Multiples Feld; Periodengruppe
 - Ausprägungen, interner Zähler, 24
 - definieren, 16
 - referenzieren, 18
 - Datenbank-Array , 20
 - Verwendung in arithmetischen Operationen, 89
- ATN, mathematische Funktion, 270
- Attribut
 - eines Feldes
 - AD-Parameter, 104
 - dynamische Zuweisung, DY-Parameter, 123
 - Konstanten, 40
- Aufrunden. *Siehe* Rundungsregeln
- Ausdrucken. *Siehe* Drucken
- Ausführen, Programm, bedingt, CC-Parameter, 110

Natural Referenzhandbuch

Ausgabe

Siehe auch Map

Datumsformat für, DFOUT-Parameter, 118

-länge eines Feldes

AL-Parameter, 108

NL-Parameter, 168

-spalten, Leerstellen zwischen, SF-Parameter, 176

-zeile

Länge von, *LOG-LS-Systemvariable, 208

Länge von, LS-Parameter, 160

zuletzt ausgegebene, *LINE-COUNT-Systemvariable, 207

Ausprägung. *Siehe* Array

Ausrichten von Feldwerten, AD-Parameter, 105

AVER-Systemfunktion, 262

B

Bedingte Programmausführung, CC-Parameter, 110

Bedingungen, logische, 45

Beenden. *Siehe* Abbrechen

Begrenzungszeichen. *Siehe* Delimiter

Benutzerdefinierte Konstanten, 30

Benutzerkennung. *Siehe* User-ID

Benutzername. *Siehe* User-Name

Benutzervariablen, 6

definieren, 8

Format/Länge, 11

FS-Parameter, 145

Berechnen. *Siehe* Arithmetik

Betriebssystem

Systemvariable *OPSYS, 213

Systemvariable *OS, 214

Systemvariable *OSVERS, 214

Bibliothek. *Siehe* Library

Bildschirm

Bedeutung von, 282

I/O, Systemvariable *SCREEN-IO, 219

Seitenlänge, Systemvariable *PAGESIZE, 215

speichern vor Window, Terminalkommando %WA, 360

Zeilenlänge, Systemvariable *LINESIZE, 207

Bildschirmfenster. *Siehe* Window

Bildschirmmaske. *Siehe* Map

Binäres Feldformat, 11
 Blinkende Feldanzeige, AD-Parameter, 105
 Boxing. *Siehe* Outlining
 Break. *Siehe* Gruppenwechsel
 BREAK-Option, in logischer Bedingung, 63
 Bruchteil eines Feldwertes, FRAC-Funktion, 270
 BX-Parameter, 109

C

C*, interner Zähler von Datenbank-Arrays, 24
 CC-Parameter, 110
 CD-Parameter, 111
 CF-Parameter, 112
 CLEAR-Taste, 291
 CO-Parameter, 113
 Compiler-Ausgabe, CO-Parameter, 113
 *CONTROL-Systemvariable, 189
 Converter-Routine, terminalspezifische, Terminalkommando %T=, 347
 *CONVID-Systemvariable, 189
 COS, mathematische Funktion, 270
 COUNT-Systemfunktion, 262
 CPU-Zeit, maximale, MT-Parameter, 166
 CTRL/D-Tasten, 291
 Cursor-Position
 auf geschützten Feldern, Terminalkommando %T+, 345
 außerhalb eines Fensters, Terminalkommando %T*, 346
 MS-Parameter, 165
 Systemvariable *CURS-COL, 191
 Systemvariable *CURS-FIELD, 191, 275
 Systemvariable *CURS-LINE, 192
 Systemvariable *CURSOR, 193
 Terminalkommando %T, 344
 CV-Parameter, 114

D

Daten
 -bank. *Siehe* Datensatz; Transaktion
 -satz. *Siehe* Datensatz
 übertragen in anderes Feld, 76
 Formatkonvertierung, 77

Natural Referenzhandbuch

- Datensatz
 - Anzahl der gelesenen, Systemvariable *NUMBER, 210
 - im Hold, WH-Parameter, 182
- Datum
 - Benutzervariablen, 14
 - Editiermaske für, 135
 - Format, DF-Parameter, 116
 - Format für Ausgabe, DFOUT-Parameter, 118
 - Format für Seitenüberschrift, DFTITLE-Parameter, 120
 - Format für Stack, DFSTACK-Parameter, 119
 - in arithmetischen Operationen, 83
 - Konstanten, 34
 - Systemvariablen, 228
- DC-Parameter, 115
- Debugging, %
- Terminalkommando, 300

- Definition, dynamischer Variablen, 233
- Delimiter, für Input-Werte, ID-Parameter, 153
- Delimiter-Modus
 - IM-Parameter, 154
 - Terminalkommando %D, 310
- Device. *Siehe* Terminal
- Dezimalzeichen, DC-Parameter, 115
- DF-Parameter, 116
- DFOUT-Parameter, 118
- DFSTACK-Parameter, 119
- DFTITLE-Parameter, 120
- *DIALOG-ID-Systemvariable, 194
- Division
 - Siehe auch* Arithmetik
 - durch Null, ZD-Parameter, 183
- Drucken
 - Hardcopy, Terminalkommando %H, 318
 - PM-Parameter, 172
 - Print-Modus, Terminalkommando %V, 351
- Drucker, für Hardcopy, Systemvariable *HARDCOPY, 198
- Dump-Generierung, DU-Parameter, 121
- DU-Parameter, 121
- Durchschnittlicher Feldwert
 - AVER-Systemfunktion, 262
 - NAVER-Systemfunktion, 263

Dynamische, Variablen und Felder, 231
 Dynamische Feldattribute
 AD-Parameter, 105
 DY-Parameter, 123
 Format C, 13
 Kontrollvariable, CV-Parameter, 114
 Dynamische Variablen, Definition, 233
 DY-Parameter, 123

E

Editiermaske
 EM-Parameter, 126
 Standardformat für Datum, DF-Parameter, 116
 Editor, Arbeitsbereich löschen, Terminalkommando %Z, 368
 Eingabe
 Siehe auch Input
 -aufforderung bei INPUT-Statement, IP-Parameter, 155
 -daten
 Anzeige unterdrücken, Terminalkommando %*, 296
 im Stack, Systemvariable *DATA, 193
 Terminalkommando, 281
 -zwang, AD-Parameter, 106
 Eingefügte Zeichen, IC-Parameter, 152
 Einteilung nach Funktion, Terminalkommandos, 286
 EJ-Parameter, 125
 EM-Parameter, 126
 End of Transaction ID. *Siehe* ETID
 ENDIAN-Modus, kompilierte Objekte, 140
 End-of-File-Bedingung, Terminalkommando %/, 298
 ES-Parameter, 141
 ETID, Systemvariable *ETID, 196
 *EVENT-Systemvariable, 197
 EXP, mathematische Funktion, 270
 Externe Subroutine. *Siehe* Subroutine

F

Farbe

des Bildschirmhintergrundes, Terminalkommando %=, 301
eines Feldes

CD-Parameter, 111

Terminalkommando %=, 301

FC-Parameter , 142

FCDP-Parameter, 143

Fehler

-meldungsanzeigeformat, Terminalkommando %MSGSF, 328

-meldungszeile, Terminalkommando %M, 326

-nummer, Systemvariable *ERROR-NR, 195

-verarbeitendes Programm, Systemvariable *ERROR-TA, 196

-verarbeitung, abschalten, Terminalkommando %E=, 314

-verursachende Sourcecodezeile, Systemvariable *ERROR-LINE, 195

- Feld
- attribute
 - AD-Parameter, 104
 - dynamische
 - CV-Parameter, 114
 - DY-Parameter, 123
 - Format C, 13
 - Ausgabelänge
 - AL-Parameter, 108
 - NL-Parameter, 168
 - format. *Siehe* Format
 - in logischer Bedingung, 72
 - interne Identifikation, POS-Funktion, 274
 - länge. *Siehe* Format
 - wert
 - absoluter, ABS-Funktion, 270
 - Anzeigeform, AD-Parameter, 104
 - Arcustangens von, ATN-Funktion, 270
 - Ausgabe identischer unterdrücken, IS-Parameter, 156
 - ausrichten, AD-Parameter, 105
 - Bruchteil von, FRAC-Funktion, 270
 - Cosinus von, COS-Funktion, 270
 - durchschnittlicher
 - AVER-Systemfunktion, 262
 - NAVER-Systemfunktion, 263
 - ganzzahliger Teil, INT-Funktion, 270
 - kleinster
 - MIN-Systemfunktion, 263
 - NMIN-Systemfunktion, 264
 - Logarithmus von, LOG-Funktion, 270
 - maximaler, MAX-Systemfunktion, 263
 - numerischer, VAL-Funktion, 270
 - Potenz von, EXP-Funktion, 270
 - Quadratwurzel von, SQRT-Funktion, 270
 - Sinus von, SIN-Funktion, 270
 - Summe
 - SUM-Systemfunktion, 264
 - TOTAL-Systemfunktion, 265
 - Tangens von, TAN-Funktion, 270
 - vor Gruppenwechsel, OLD-Systemfunktion, 264
 - Vorzeichen von, SGN-Funktion, 270
 - Felder, große und dynamische, 231

Natural Referenzhandbuch

Fenster

Siehe auch Window

Bedeutung von, 282

Fließkomma. *Siehe* Gleitkomma

Fließpunkt. *Siehe* Gleitkomma

Floating Point. *Siehe* Gleitkomma

FL-Parameter, 144

Format

Konvertierung von Daten in ein anderes, 77

Performance-Hinweise für Arithmetik, 86

prüfen in logischer Bedingung, 65

von Benutzervariablen, 11

FS-Parameter, 145

FORMAT-Statement, 99

Forms/Screen-Modus, Terminalkommando %F, 315

Forms-Modus

IM-Parameter, 154

Terminalkommando %F, 314

Fortsetzungszeichen %, 292

FRAC, mathematische Funktion, 270

FS-Parameter, 145

Füllzeichen

für dynamisch geschützte Eingabefelder, FCDP-Parameter, 143

für Felder, AD-Parameter, 107

für Spaltenüberschriften

FC-Parameter , 142

GC-Parameter, 146

Funktionen

mathematische Funktionen, 270

Systemfunktionen, 259

Funktionstasten

-leiste

- KD-Parameter, 157

- Terminalkommando %Y, 365

Siemens-Logik

- Terminalkommando %KN, 323

- Terminalkommando %KO, 323

- Terminalkommando %KS, 323

simulieren

- Terminalkommando %Knn, 324

- Terminalkommando %KPn, 324

zuletzt gedrückte

- Systemvariable *PF-KEY, 217

- Systemvariable *PF-NAME, 218

G

Ganzzahl, Format I, 11

Ganzzahliger Teil eines Feldwertes, INT-Funktion, 270

GC-Parameter, 146

Generieren, Symboltabelle, 179

Gepackt numerisch, Format P, 11

Gerätetyp. *Siehe* Terminaltyp

Geschütztes Feld

- AD-Parameter, 106

- Cursor auf, Terminalkommando %T+, 345

- überschreiben durch Helproutine verhindern, OPF-Parameter, 169

Gleitkomma

- Format F, 11

- in arithmetischen Operationen, 81

- Konstanten, 39

- Mantissenlänge, FL-Parameter, 144

GLOBALS-Systemkommando, 98

Großbuchstaben, Umsetzung in

- AD-Parameter, 107

- Terminalkommando %L, 325

- Terminalkommando %U, 350

Große, Variablen und Felder, 231

Großrechner, Begriffsdefinition, 3

Gruppe, in Natural Security, Systemvariable *GROUP, 197

Natural Referenzhandbuch

Gruppenwechsel
 Feldwert vor, OLD-Systemfunktion, 264
 in logischer Bedingung, 63
GUI-Handle. *Siehe* Handle

H

Handle, 15
 in logischer Bedingung, 48
 -Konstanten, 40
Hardcopy
 Terminalkommando %H, 318
 zugewiesener Drucker, Systemvariable *HARDCOPY, 198
HC-Parameter, 146
Help. *Siehe* Hilfe
Helproutine
 aufrufen, Terminalkommando %J, 322
 überschreiben geschützter Felder verhindern, OPF-Parameter, 169
 zuweisen, HE-Parameter, 147
HE-Parameter, 147
Hexadezimaler Feld
 Editiermaske für, 133
 Konstante, 37
Hilfe, für Terminalkommandos, Terminalkommando %?, 282
Hintergrundfarbe, Terminalkommando %=, 301
Höchster Feldwert, MAX-Systemfunktion, 263
Hold, Datensatz im, WH-Parameter, 182
HW-Parameter, 150

I

IA-Parameter, 151
IC-Parameter, 152
ID. *Siehe* Library-ID; Terminal-ID; User-ID
Identical Suppress, IS-Parameter, 156
Identische Feldwerte unterdrücken. *Siehe* Identical Suppress
ID-Parameter, 153
IM-Parameter, 154
Index. *Siehe* Array
Infoline, Terminalkommando %X, 361
Inline-Subroutine. *Siehe* Subroutine

Input

- Delimiterzeichen, ID-Parameter, 153
- Modus, IM-Parameter, 154
- Zuweisungszeichen, IA-Parameter, 151

INPUT-Statement

- Ausführung wiederholen, Terminalkommando %R, 338
- Eingabeaufforderung, IP-Parameter, 155
- Fortsetzungszeichen %, 292
- Verarbeitung unterdrücken, SET CONTROL 'Q', 336

INT, mathematische Funktion, 270

Integer. *Siehe* Ganzzahl

Intensivierte Anzeige von Feldern, AD-Parameter, 105

Interne Satznummer. *Siehe* ISN

Interne Subroutine. *Siehe* Subroutine

Interner Zähler von Datenbank-Arrays, C*, 24

Inverse Anzeige von Feldern, AD-Parameter, 105

IP-Parameter, 155

ISN, Systemvariable *ISN, 202

IS-Option, in logischer Bedingung, 65

IS-Parameter, 156

J

Jahr 2000

- Datumsformat, DF-Parameter, 116
- Datumsformat für Ausgabe, DFOUT-Parameter, 118
- Datumsformat für Seitenüberschrift, DFTITLE-Parameter, 120
- Datumsformat für Stack, DFSTACK-Parameter, 119

K

KD-Parameter, 157

Kennung. *Siehe* ID

Kennzeichen. *Siehe* Zeichen

Keyword/Delimiter-Modus

- IM-Parameter, 154
- Terminalkommando %D, 310

Kleinbuchstaben, Umsetzung in Großbuchstaben

- AD-Parameter, 107
- Terminalkommando %L, 325
- Terminalkommando %U, 350

Natural Referenzhandbuch

- Kleinster Feldwert
 - MIN-Systemfunktion, 263
 - NMIN-Systemfunktion, 264
- Kommandos, Terminalkommandos, 279
- Kommentar, im Sourcecode, 44
- Kommunikationsbereich, *COM-Systemvariable, 188
- Kompilierte Objekte, ENDIAN-Modus, 140
- Konstante, 30
 - alphanumerische, 32
 - Attribut-, 40
 - für Datum, 34
 - für Zeit, 34
 - Gleitkomma-, 39
 - Handle-, 40
 - hexadezimale, 37
 - logische, 38
 - numerische, 30
- Kontrollvariable, 13
 - auf Veränderung prüfen, 69
 - CV-Parameter, 114
- Konvertierung von Daten in anderes Format, 77
- Konvertierung von Systemdatei-Ausgaben, TS-Parameter, 180
- Kosinus eines Feldes, COS-Funktion, 270
- Kursive Anzeige von Feldwerten, AD-Parameter, 105

L

- Label
 - zur Qualifizierung von Datenstrukturen, 28
 - zur Statement-Referenzierung, 9
- Länge. *Siehe* Format
- Länge, von Feld oder Variable, Systemvariable *LENGTH, 206
- LC-Parameter, 158
- LE/370, Aufruf von dynamischem Hauptprogramm, Terminalkommando %P=L, 335
- Leerzeichen zwischen Ausgabespalten, SF-Parameter, 176
- Leerzeilen, unterdrücken, ES-Parameter, 141
- LE-Parameter, 159
- Level, des gerade ausgeführten Programms, Systemvariable *LEVEL, 206

- Library
 - für Recording, Terminalkommando %B=, 306
 - ID
 - Systemvariable *APPLIC-ID, 187
 - Systemvariable *LIBRARY-ID, 206
 - Name, Systemvariable *APPLIC-NAME, 187
 - verknüpfte. *Siehe* Steplib
- Lichtstift, Terminalkommando %RM, 339
- Limit, für Verarbeitungsschleife
 - LE-Parameter, 159
 - LT-Parameter, 161
- Linksbündige Feldwertausgabe, AD-Parameter, 105
- LOG, mathematische Funktion, 270
- Logarithmus eines Feldwertes, LOG-Funktion, 270
- Logische Bedingung, 45
 - arithmetischer Ausdruck, 48
 - Auswertung einer logischen Variablen, 67
 - BREAK-Option, 63
 - erweiterter relationaler Ausdruck, 51
 - Felder in, 72
 - Handle, 48
 - IS-Option, 65
 - MASK-Option, 52
 - modifizierte Kontrollvariable, 69
 - relationaler Ausdruck, 47
 - SCAN-Option, 61
 - Verknüpfen von, 74
- Logisches Feld
 - Editiermaske für, 138
 - Format L, 15
 - in logischer Bedingung, 67
 - Konstante, 38
- Loop. *Siehe* Verarbeitungsschleife
- LÖSCH-Taste. *Siehe* CLEAR-Taste
- LS-Parameter, 160
- LT-Parameter, 161

M

- Mantissenlänge, FL-Parameter, 144

Natural Referenzhandbuch

Map

Siehe auch Ausgabe

Ausgabe unterdrücken, Terminalkommando %Q, 336

Ausgabe verzögern, Terminalkommando %QS, 337

Maske

Siehe auch Editiermaske; Map

MASK-Option, in logischer Bedingung, 52

MASK-Option, in logischer Bedingung, 52

Mathematische Berechnungen. *Siehe* Arithmetik

Mathematische Funktionen, 270

MAX-Systemfunktion, 263

MC-Parameter, 162

Meldungszeile

Position, 163

Terminalkommando %M, 326

Message Line. *Siehe* Meldungszeile

MIN-Systemfunktion, 263

Modifizierbares Feld, AD-Parameter, 106

Modifizierte Kontrollvariable, 69

MP-Parameter, 164

MS-Parameter, 165

MT-Parameter, 166

Multiples Feld

Siehe auch Array

Anzahl ausgegebener Werte, MC-Parameter, 162

Multiplikation. *Siehe* Arithmetik

N

Nachgestellte Zeichen, TC-Parameter, 180

NATPAGE-Utility

Systemdatei, PD-Parameter, 171

Terminalkommando %E, 312

Terminalkommando %I, 321

Terminalkommando %O, 330

Terminalkommando %P, 331

Terminalkommando %S, 343

Natural Connection, Terminalkommandos %+ und %-, 299

NAVER-Systemfunktion, 263

NCOUNT-Systemfunktion, 263

NC-Parameter, 167

Nichtangezeigter Feldwert, AD-Parameter, 105

NL-Parameter, 168
 NMIN-Systemfunktion, 264
 Non-Conversational-Modus, Terminalkommando %N, 329
 Null
 Auffüllen eines Feldes mit Nullen, AD-Parameter, 105
 Ausgabe von, ZP-Parameter, 184
 Division durch, ZD-Parameter, 183
 Numerierung von Sourcecode-Zeilen, Referenzierung über, 9
 Numerischer Wert eines alphanumerischen Feldes, VAL-Funktion, 270
 Numerisches Feld
 Ausgabelänge, NL-Parameter, 168
 Editiermaske für, 128
 Format N, 11
 gepackt, Format P, 11
 Konstante, 30

O

Objekt. *Siehe* Programm
 Objekt-Handle. *Siehe* Handle
 OLD-Systemfunktion, 264
 OpenVMS, Begriffsdefinition, 3
 Operation, abbrechen, Terminalkommandos % und %%, 293
 OPF-Parameter, 169
 Outlining
 BX-Parameter, 109
 Terminalkommando %D=, 310

P

Page Buffer, kopieren, Terminalkommando %C, 307
 Page Dataset. *Siehe* NATPAGE-Utility
 Parameter. *Siehe* Session-Parameter
 Parameterdatei, aktuell benutzte, Systemvariable *PARM-USER, 216
 Paßwort, Anzeige unterdrücken, Terminalkommando %*, 296
 PA-Tasten. *Siehe* Funktionstasten
 Patch-Level-Nummer, als String-Wert, Systemvariable *PATCH-LEVEL, 216
 PC-Parameter, 170
 PD-Parameter, 171
 Periodengruppe
 Siehe auch Array
 Anzahl ausgegebener Ausprägungen, PC-Parameter, 170

Natural Referenzhandbuch

PF-Tasten. *Siehe* Funktionstasten
PM-Parameter, 172
POS, Feldidentifikationsfunktion, 274
Potenzierung
 Siehe auch Arithmetik
 EXP-Funktion, 270
Print-Modus
 PM-Parameter, 172
 Terminalkommando %V, 351
Profilparameter
 DSLMM, 235
 USIZE, 235
Programm
 ausführen, bedingt, CC-Parameter, 110
 gerade ausgeführtes
 Level, Systemvariable *LEVEL, 206
 Systemvariable *INIT-PROGRAM, 200
 Systemvariable *PROGRAM, 218
Programmiermodus, SM-Parameter, 178
Prozeß-ID, als String-Wert, Systemvariable *PID, 218
Prüfungen, Speichergrenzen, 235
Pseudo-Dialogbetrieb, Ausgabe unterdrücken, Terminalkommando %QO, 336
PS-Parameter, 174

Q

Quadratwurzel eines Feldwertes, SQRT-Funktion, 270

R

Rechnen. *Siehe* Arithmetik
Rechtsbündige Feldwertausgabe, AD-Parameter, 105
Recording
 abspielen, Terminalkommando %A, 304
 Abspielmodus, Terminalkommando %G, 317
 Abspielung ändern, Terminalkommando %R, 338
 aufzeichnen, Terminalkommando %B, 305
 Helproutine aufrufen, Terminalkommando %J, 322
 Library für, Terminalkommando %B=, 306
 unterbrechen, CLEAR-Taste, 291
Referenzen, auf Sourcecode-Zeilennummern, umnumerieren, 96

Referenzierung
 von Arrays, 18
 Datenbank-Arrays, 20
 interner Zähler C*, 24
 von Statements, 9
 REINP-Parameter, 175
 REINPUT-Statement, internes bei ungültigen Daten, REINP-Parameter, 175
 Relationaler Ausdruck, in logischer Bedingung, 47, 51
 Remote Procedure Call. *Siehe* RPC
 Report
 maximale Länge, MP-Parameter, 164
 -Spezifikation, 41
 Reporting Mode. *Siehe* Programmiermodus
 RET, Returncode-Funktion, 276
 Returncode, eines Nicht-Natural-Programms, RET-Funktion, 276
 RPC, Konversations-ID, Systemvariable *CONVID, 189
 Rückmeldungszeile, Terminalkommando %M, 326
 Rundungsregeln, 79

S

SA-Parameter, 175
 Satznummer, Systemvariable *ISN, 202
 Scannen, in einem Feld, SCAN-Option in logischer Bedingung, 61
 SCAN-Option, in logischer Bedingung, 61
 Schirm. *Siehe* Bildschirm; Map
 Schleife. *Siehe* Verarbeitungsschleife
 Schutz eines Feldes, temporär, AD-Parameter, 106
 Screen Paging. *Siehe* NATPAGE-Utility
 Screen-Modus, Terminalkommando %F, 314
 Seite
 gerade ausgegebene, Systemvariable *PAGE-NUMBER, 215
 logische Länge
 PS-Parameter, 174
 Systemvariable *LOG-PS, 208
 maximale Anzahl eines Reports, MP-Parameter, 164
 physische Länge, Systemvariable *PAGESIZE, 215
 Seitenpuffer kopieren, Terminalkommando %C, 307
 Seitenvorschub, EJ-Parameter, 125
 Überschrift, DFITITLE-Parameter, 120
 Semikolon, als Statement-Ende, 44
 Server, Typ, Systemvariable *SERVER-TYPE, 220

Natural Referenzhandbuch

- Session-Parameter, 97
 - Standardwerte ändern, 98
- SF-Parameter, 176
- SGN, mathematische Funktion, 270
- SG-Parameter, 177
- SIN, mathematische Funktion, 270
- Sinus eines Feldwertes, SIN-Funktion, 270
- SL-Parameter, 177
- SM-Parameter, 178
- Sortierschlüssel, SORTKEY-Funktion, 277
- SORTKEY, Sortierschlüssel-Funktion, 277
- Sourcecode
 - Kommentar im, 44
 - Zeile, maximale Länge, SL-Parameter, 177
 - Zeilennummern, zur Statement-Referenzierung, 9
- Sourcecode-Zeilenummern, Referenzen auf, umnumerieren, 96
- Spaltenüberschrift
 - ausrichten, HC-Parameter, 146
 - Breite, HW-Parameter, 150
- SPECIFIED-Option, 71
- Speichergrenzen, Prüfungen, 235
- Spezifikation von Reports, 41
- Sprachindikator
 - Systemvariable *LANGUAGE, 203
 - Terminalkommando %L=, 325
- SQL
 - Anzahl der verarbeiteten Zeilen, Systemvariable *ROWCOUNT, 219
 - Datenbanken, Begriffsdefinition, 3
- SQRT, mathematische Funktion, 270
- Stack
 - Daten ablegen, Terminalkommando %CS, 308
 - Datumsformat für, DFSTACK-Parameter, 119
 - Eintrag lesen ohne zu löschen, Terminalkommando %.S, 297
 - Inhalt des, Systemvariable *DATA, 193
 - obersten Eintrag löschen, Terminalkommando %.P, 297
- Startup-Programm, Systemvariable *STARTUP, 221
- Statements
 - EXPAND, 236
 - REDUCE, 236
 - Referenzierung, 9
 - Syntaxsymbole, 2
- Statistikzeile, Terminalkommando %X, 361

Steplib, Systemvariable *STEPLIB, 222
Steuerzeichen
 Siehe auch Zeichen
 für Terminalkommandos, 280
Structured Mode. *Siehe* Programmiermodus
Stufe. *Siehe* Level
Subroutine, gerade ausgeführte, Systemvariable *SUBROUTINE, 223
SUBSTRING-Option, relationaler Ausdruck, 49
Subtraktion. *Siehe* Arithmetik
Summe, von Werten eines Feldes
 SUM-Systemfunktion, 264
 TOTAL-Systemfunktion, 265
SUM-Systemfunktion, 264
Symboltabelle, generieren, 179
Syntaxsymbole, 2
Systemfunktionen, 259
Systemkommandos, verbieten, NC-Parameter, 167
Systemmeldungszeile, Terminalkommando %M, 326
Systemvariable, *LENGTH(Feld), 234
Systemvariablen, 185

T

Tabelle. *Siehe* Array
TAN, mathematische Funktion, 270
Tangens eines Feldwertes, TAN-Funktion, 270
Tasten. *Siehe* Funktionstasten
Tastenbelegungen, Terminalkommandos, 290
TC-Parameter, 180
Technische Informationen, %
Terminalkommando, 300

Teilung. *Siehe* Division
Teleprocessing Monitor. *Siehe* TP-Monitor
Temporärer Schutz eines Feldes, AD-Parameter, 106

Natural Referenzhandbuch

Terminal

- ID, Systemvariable *INIT-ID, 199
- kommandos, 279
 - Steuerzeichen, CF-Parameter, 112
- Seitenlänge, Systemvariable *PAGESIZE, 215
- typ
 - Converter-Routine, Terminalkommando %T=, 347
 - Systemvariable *DEVICE, 194
- warnton, SA-Parameter, 175
- Zeilenlänge, Systemvariable *LINESIZE, 207

Terminalkommandos

- nach Funktionen eingeteilt, 286
- Tastenbelegungen, 290

Text-Notation, 42

- Time-out, Systemvariable *TIME-OUT, 229
- Timestamp, Systemvariable *TIMESTAMP, 229
- TOTAL-Systemfunktion, 265
- TP-Monitor, Systemvariable *TPSYS, 224

Trace

- extern, Terminalkommando %TRE, 348
- intern, Terminalkommando %TRI, 349

Transaktion

- abbrechen, bei Zeitüberschreitung, Systemvariable *TIME-OUT, 229
- Transaktionsdaten identifizieren, Systemvariable *ETID, 196

TS-Parameter, 180

U

- UC-Parameter, 181
- Uhrzeit. *Siehe* Zeit
- Umnummerieren, Sourcecode-Zeilenummern, Referenzen auf, 96
- Ungeschütztes Feld, AD-Parameter, 106
- UNIX, Begriffsdefinition, 3
- Unterbrechen. *Siehe* Abbrechen
- Unterstreichen, von Feldwerten, AD-Parameter, 105
- Unterstreichungszeichen, UC-Parameter, 181
- User
 - ID
 - Systemvariable *GROUP, 197
 - Systemvariable *INIT-USER, 200
 - Systemvariable *USER, 225
 - Name, Systemvariable *USER-NAME, 225

V

- VAL, mathematische Funktion, 270
 - Formatprüfung, 65
- Variablen
 - Siehe auch* Benutzervariablen; Systemvariablen
 - Definition dynamischer Variablen, 233
 - große und dynamische, 231
- Verarbeitungsschleife
 - abbrechen, RESET/ENTER-Tasten, 291
 - Anzahl der Durchläufe
 - begrenzen, LT-Parameter, 161
 - Limit-Überschreitung, LE-Parameter, 159
 - zählen
 - COUNT-Systemfunktion, 262
 - NCOUNT-Systemfunktion, 263
 - Systemvariable *COUNTER, 190
- Vorangestellte Nullen, AD-Parameter, 105
- Vorangestellte Zeichen, LC-Parameter, 158
- Vorzeichen
 - eines Feldes, SG-Parameter, 177
 - eines Feldwertes, SG-Funktion, 270

W

- Warnton, SA-Parameter, 175
- WH-Parameter, 182
- Window
 - aktives Feld außerhalb von, *COM-Systemvariable, 188
 - Bildschirm speichern vor, Terminalkommando %WA, 360
 - Cursor außerhalb von, Terminalkommando %T*, 346
 - Outlining, Terminalkommando %D=, 310
 - Position, Systemvariable *WINDOW-POS, 226
 - Rahmenzeichen, Terminalkommando %F=, 316
 - Seitenlänge, Systemvariable *WINDOW-PS, 226
 - Steuerung, Terminalkommando %W, 352
 - Zeilenlänge, Systemvariable *WINDOW-LS, 225
- Windows, Begriffsdefinition, 3
- WSIZE-Profilparameter, 360
- Wurzel. *Siehe* Quadratwurzel

Z

ZD-Parameter, 183

Zeichen

eingefügte, IC-Parameter, 152

Fortsetzungszeichen %, 292

für Dezimalkomma, DC-Parameter, 115

für Terminalkommandos, CF-Parameter, 112

für Input-Delimiter, ID-Parameter, 153

für Input-Zuweisung, IA-Parameter, 151

für Unterstreichung, UC-Parameter, 181

nachgestellte, TC-Parameter, 180

vorangestellte, LC-Parameter, 158

zum Auffüllen von Feldern, AD-Parameter, 107

zum Auffüllen von Spaltenüberschriften

FC-Parameter , 142

GC-Parameter, 146

zum Füllen von Eingabefeldern, unterdrücken von, FCDP-Parameter, 143

Zeile. *Siehe* Ausgabezeile; Sourcecode-Zeile

Zeit

Benutzervariablen, 14

Editiermaske für, 135

hardware-interner Zähler, Systemvariable *TIMESTMP, 229

in arithmetischen Operationen, 83

Konstanten, 34

maximale CPU-Zeit, MT-Parameter, 166

Systemvariablen, 228

-überschreitung bei Datenbankzugriff, Systemvariable *TIME-OUT, 229

Zentrieren, von Spaltenüberschriften, HC-Parameter, 146

ZP-Parameter, 184

